Tackling the Data Bottleneck and Inference Challenges in Machine Learning

by

Harit Vishwakarma

A dissertation submitted in partial fulfillment
of the requirements for the degree of

Doctor of Philosophy
(Computer Sciences)

at the
University of Wisconsin-Madison
2025

Date of Final Oral Exam: 04/25/2025

The dissertation is approved by the following members of the Final Oral Committee:

Frederic Sala, Assistant Professor, Computer Sciences,
Ramya Korlakai Vinayak, Assistant Professor, Electrical & Computer Engineering,
Yudong Chen, Associate Professor, Computer Sciences,
Robert Nowak, Professor, Electrical & Computer Engineering,
Xiaojin (Jerry) Zhu, Professor, Computer Sciences.

© Copyright by Harit Vishwakarma 2025
All Rights Reserved

Dedicated to my beloved family.

Acknowledgments

This thesis marks the culmination of a transformative journey over the past six years, and I am deeply grateful to all those who have supported me along the way.

Foremost, I wish to thank my advisors, Prof. Frederic Sala and Prof. Ramya Korlakai Vinayak, for their invaluable guidance, time, and unwavering support throughout my PhD. Their advice on technical and non-technical aspects has shaped not only this work but also the researcher I have become. It has been a deeply fulfilling experience to be part of their vibrant and diverse research groups, and I am fortunate to have had this unique opportunity.

I am grateful to the members of my thesis committee—Prof. Yudong Chen, Prof. Robert Nowak, and Prof. Xiaojin (Jerry) Zhu—for their time, insightful feedback, and discussions. I also thank Prof. Dimitris Papailiopoulos and Prof. Kangwook Lee for the collaboration and advice during the earlier part of my Ph.D. Beyond research, I have been fortunate to learn the fundamentals of machine learning from courses taught by exceptional instructors. These courses not only equipped me with the necessary skills and knowledge to pursue research, but also inspired me deeply. In particular, I would like to thank Prof. Robert Nowak, Prof. Jerry Zhu, Prof. Ilias Diakonikolas, Prof. Steve Wright, Prof. Jelena Diakonikolas, and Prof. Shivaram Venkataraman for their outstanding courses on theoretical machine learning, nonlinear optimization, and ML systems.

This thesis would not have been possible without the brilliant collaborators and labmates I had the pleasure of working with. I owe a great debt of gratitude to Heguang Lin, Yi Chen, Daisuke Yamada, Tzu-Heng Huang, Namburi GNVV Satya Sai Srinath, Sui Jiet Tay, Changho Shin, and Nick Roberts for their contributions, support, and many enjoyable collaborations along the way. I also thank Winfred Li, Jitian Zhao, Sonia Cromp, Dyah Adila, Kendall Park, Gokcan Tatli, Daiwei Chen, Aniket Rege, Zhi Wang and the rest of the labmates for fruitful discussions and feedback that helped me during this journey. I extend my gratitude to Ankit Pensia, Shashank Rajput, Hongyi Wang, Kartik Srinivasan, Saurabh Agarwal, Jy-yong Sohn and Greg Canal with whom I had a pleasure to collaborate and learn from.

Part of the work in the thesis was completed during an internship at J.P. Morgan AI Research. I am grateful to Alan Mishler, Niccolò Dalmasso, Thomas Cook, Natraj Raman, and Sumitra Ganesh for the collaboration. I also thank Sujay Bhatt, Alec Koppel, and other friends at J.P. Morgan for the discussions, feedback, and fun activities.

I am grateful to the Department of Computer Sciences, the Department of Electrical and Computer Engineering, and the broader machine learning community at the University of Wisconsin–Madison for providing the resources and a stimulating academic environment that greatly helped in my

research. I sincerely thank American Family Insurance and the Institute for Foundations of Data Science (IFDS) for supporting my work. I also appreciate the organizers of the SILO, MLOPT, and IFDS seminars, which gave me the opportunity to learn from leading researchers around the world and to present my work and receive valuable feedback. I thank the anonymous reviewers for their thoughtful and constructive feedback, which helped improve the quality of my papers.

Beyond academics, I was fortunate to share my time in Madison with wonderful friends who made the journey memorable and full of joy. I feel incredibly lucky to have shared this time with Bhavya Kenia, Nachiket Kshatriya, Dwijesh Kalyanaraman, Daria Tennikova, Darti, and Barbie, with whom I have made lifelong memories—from bittersweet board games, spicy curries, concerts, and badminton, to deep conversations about life, the universe, and everything. I have to say, the dice must have rolled in my favor, and I found these sheep and the two shepherds here.

Being part of the two labs and the vibrant student community, I was lucky to cherish this journey with several friends: Nick Roberts, Tzu-Heng Huang, Yi Chen, Heguang Lin, Dyah Adila, Gokcan Tatli, Aniket Rege, Daisuke Yamada, Changho Shin, Sonia Cromp, Jitian Zhao, Namburi GNVV Satya Sai Srinath, Angeliki Giannou, Daiwei Chen, Zhi Wang, Rheeya Uppal, Joe Shenouda, Subhojyoti Mukherjee, Greg Canal, Kartik Sreenivasan, Ashish Hooda, Shubham Bharti, Sourav Pal, Harsha Adepu, and Varun Sundar. Thank you for all the fun and conversations we had.

I really appreciate the excellent sporting facilities at the university, especially the Nicholas Recreation Center and its staff. The numerous hours spent in badminton, swimming, running, and enjoying racquetball have helped me reset and maintain my fitness and connect with people. In addition, I wish to thank the city of Madison for providing excellent paths for biking and running. It has been a luxury to commute to work and most places of interest to me with a short and leisurely bike ride—even through Wisconsin winters. With its calm atmosphere, scenic lakes, and kind people, Madison has been a wonderful place to live and grow, both personally and academically.

My gratitude extends to the faculty and colleagues at IISc Bangalore and IBM Research India for the mentorship, teaching, and research environment that inspired me to pursue a Ph.D. I would especially like to thank Prof. Chiranjib Bhattacharyya, Prof. Shirish Shevade, Prof. Ambedkar Dukkipati, Sumit Bhatia, L. Venkata Subramaniam, and Dinesh Garg for their guidance and support during my time as a master's student at IISc and while working at IBM Research.

Most of all, I am forever grateful to my parents, Satyanarayan Vishwakarma & Mamta Vishwakarma, my younger brother Amit Vishwakarma, and my younger sister Raksha Vishwakarma for their unwavering love and support. It speaks of their incredible will, sacrifices, and hard work that, despite financial hardships and very little education, they always supported my decisions and the education of three of us.

Abstract

The landscape of machine learning (ML) is ever-expanding and opening up new avenues of application. Yet, the path to building a successful ML application is far from straightforward and riddled with two critical challenges: acquiring sufficient high-quality labeled data for training (or fine-tuning) and ensuring robust performance when trained models are deployed in real-world environments. This thesis addresses both challenges through complementary research directions that span the ML pipeline from data preparation to inference.

The first part of this thesis investigates and improves solutions aimed at alleviating the labeled data bottleneck. We focus on threshold-based auto-labeling (TBAL), pseudolabeling-based semi-supervised learning, also known as self-training, and the broader framework on weak supervision (WS). We make the following contributions spanning across these paradigms,

- **Foundations and Limits of Threshold-Based Auto-Labeling (TBAL).** We begin with a foundational study of TBAL, a widely used technique that promises to label large portions of unlabeled data while keeping the error rate below a user-specified tolerance ϵ_a. Despite its practical success and growing adoption, TBAL remained theoretically underexplored. We present the first rigorous theoretical and empirical analysis of TBAL, establishing conditions under which it fulfills its promises. Our results show that TBAL can yield accurate labels even with moderate-quality models, but also reveal a key limitation: in worst-case scenarios, the required validation sample size scales as $\mathcal{O}(1/\epsilon_a^2)$. Such requirements can undermine the utility of TBAL in practice.

- **Optimizing Confidence Functions for TBAL.** We then address a critical bottleneck in TBAL—the reliance on ad-hoc confidence scores—by proposing an optimization framework (Colander) that learns confidence functions aligned with the TBAL objective. Our experiments show that using Colander in TBAL boosts the number of auto-labeled points by up to 60% while keeping the errors below 5%.

- **Improvements to Pseudolabeling-Based Semi-Supervised Learning.** Extending our insights from TBAL, we tackle similar challenges in pseudolabeling-based semi-supervised learning (SSL) (self-training), where pseudolabeling relies on ad-hoc confidence scores and thresholds. We develop a principled framework to learn both scores and thresholds with explicit error control, improving the quality and quantity of pseudolabels. When integrated into modern SSL pipelines, this approach yields significant gains in accuracy and training efficiency.

- **Advancing Weak Supervision for Structured Prediction.** Finally, we advance WS for structured prediction tasks, introducing novel techniques based on pseudo-Euclidean embeddings and tensor decomposition for label spaces such as finite discrete metric spaces and constant-curvature Riemannian manifolds, with generalization guarantees matching those of clean-label settings.

The second part of this thesis focuses on statistical inference techniques to ensure safe and reliable model deployment in dynamic and high-stakes settings. We make the following contributions,

- **Human-in-the-Loop Out-of-Distribution Detection with FPR Guarantees**. First, we address the challenge of out-of-distribution (OOD) detection, where models trained on in-distribution (ID) (positive) data must handle OOD (negative) inputs at test time. We propose a human-in-the-loop framework that adapts both the scoring function and decision thresholds in response to real-world OOD inputs, while providing formal guarantees on false positive rate (FPR) control at all times. This method ensures high true positive rates (TPR) without sacrificing safety and significantly outperforms existing baselines on the OpenOOD benchmarks.

- **Improving LLM Inference with Conformal Prediction.** Second, we investigate how conformal prediction (CP)—a statistical framework that produces prediction sets with formal coverage guarantees—can help large language models (LLMs) on structured tasks such as multiple-choice questions (MCQs) and tool usage. We introduce Conformal Revision of Questions (CROQ), a method inspired by the Monty Hall problem and human test-taking

strategies, which uses CP to prune distractor choices and re-prompts the LLM with the revised, narrowed-down question. To further enhance CROQ's effectiveness, we propose CP-OPT, an optimization framework that learns task-specific scoring functions aimed at producing smaller prediction sets while maintaining coverage. These smaller sets benefit not only CROQ by making revised questions more focused, but also uncertainty-aware pipelines that defer or abstain based on set size. Together, CROQ and CP-OPT provide a principled approach to improving both the accuracy and reliability of LLM outputs at test time.

Through these contributions, this thesis advances both the efficiency of labeled data acquisition and the robustness of machine learning models in real-world deployments, while setting the stage for future research on enhancing data labeling pipelines and improving the quality and reliability of model inferences in dynamic environments.

Contents

1 Organization **2**

I Tackling the Data Bottleneck **5**

2 Overview **6**
2.1 Motivation . . . 6
2.2 Literature . . . 8
2.3 Contributions . . . 10
2.3.1 Understanding and Improving Threshold-based Auto-labeling . . . 10
2.3.2 Improving Pseudolabeling in Semi-Supervised Learning . . . 12
2.3.3 Extending Weak Supervision to Structured Prediction. . . . 12
2.4 Background . . . 13
2.4.1 Threshold-based Auto-labeling . . . 13
2.4.2 Weak Supervision . . . 15
2.5 Outlook . . . 17

3 Threshold-based Auto-labeling **19**
3.1 Introduction . . . 19
3.2 Threshold-Based Auto-Labeling Algorithm . . . 22
3.2.1 Problem Setup . . . 22
3.2.2 Description of the Algorithm . . . 23
3.2.3 Comparison between Auto-Labeling, Active Learning and Selective Classification . . . 25
3.3 Theoretical Analysis . . . 26
3.3.1 Linear Classifier Setting . . . 30
3.4 Experiments . . . 32
3.4.1 Role of Validation Data . . . 35
3.4.2 Role of Training Data Size . . . 36
3.5 Related Works . . . 37
3.6 Conclusions and Future Work . . . 39

4 Confidence Functions for Auto-labeling **40**
4.1 Introduction . . . 40
4.2 Background and Motivation . . . 43

4.3 Proposed Method (`Colander`) . . . 46
4.3.1 Auto-labeling optimization framework . . . 47
4.3.2 Practical method to learn confidence functions . . . 47
4.3.3 TBAL procedure with `Colander` . . . 50
4.4 Empirical Evaluation . . . 52
4.4.1 Baselines . . . 52
4.4.2 Datasets and models . . . 53
4.4.3 Hyperparameter search and evaluation . . . 53
4.4.4 Results and discussion . . . 54
4.5 Related Work . . . 56
4.6 Conclusion . . . 57

5 Confidence Scores and Thresholds in Pseudolabeling-based SSL 59
5.1 Introduction . . . 60
5.2 Background and Problem Setup . . . 62
5.2.1 Pseudolabeling-based Semi-Supervised Learning . . . 62
5.2.2 Quality and Quantity of Pseudolabels . . . 64
5.2.3 Our Goals . . . 64
5.3 Methodology . . . 65
5.3.1 Pseudolabeling Optimization Framework . . . 65
5.3.2 Threshold Estimation . . . 67
5.3.3 Pseudolabeling and Accumulation . . . 68
5.4 Experiments . . . 69
5.4.1 Experimental Setup . . . 70
5.4.2 Results and Discussion . . . 71
5.5 Related Work . . . 75
5.6 Conclusion . . . 76

6 Weak Supervision and Structured Prediction 78
6.1 Introduction . . . 79
6.2 Background and Problem Setup . . . 81
6.2.1 Structured Prediction . . . 81
6.2.2 Weak Supervision . . . 82
6.3 Noise Rate Recovery in Finite Metric Spaces . . . 84
6.3.1 Pseudo-Euclidean Embeddings . . . 85
6.3.2 Multi-View Mixtures and Tensor Decompositions . . . 86
6.4 Generalization Error for Structured Prediction in Finite Metric Spaces . . . 89
6.4.1 Prediction with Pseudolabels . . . 89
6.4.2 Bounding the Generalization Error . . . 90
6.5 Manifold-Valued Label Spaces: Noise Recovery and Generalization . . . 91
6.6 Experiments . . . 94
6.7 Conclusions . . . 96

II Improving Reliability and Accuracy at Inference-time **99**

7 Overview **100**
7.1 Motivation . . . 100
7.2 Literature . . . 101
7.3 Contributions . . . 103
7.3.1 Anytime Robustness to Out-Of-Distribution Points . . . 103
7.3.2 Improving Language Model Inference with Conformal Prediction . . . 104
7.4 Background . . . 106
7.4.1 Safe Anytime Valid Inference (SAVI) . . . 106
7.4.2 Conformal Inference . . . 107
7.5 Outlook . . . 108

8 Anytime Robustness to Out-Of-Distribution Points I **110**
8.1 Introduction . . . 111
8.2 Human-in-the-Loop OOD Detection . . . 113
8.2.1 Problem Setting . . . 113
8.2.2 Adaptive Threshold Estimation . . . 115
8.3 Theoretical Guarantees . . . 118
8.4 Empirical Evaluation . . . 121
8.4.1 Stationary Distributions Setting . . . 122
8.4.2 Distribution Shift Setting . . . 123
8.5 Related Works . . . 126
8.6 Conclusions . . . 129

9 Anytime Robustness to Out-of-Distribution Points II **130**
9.1 Introduction . . . 131
9.2 Preliminaries . . . 133
9.2.1 Problem Setting . . . 133
9.3 Methodology . . . 134
9.3.1 ASAT Workflow . . . 135
9.3.2 Learning Scoring Functions and Thresholds . . . 135
9.3.3 Learning Thresholds . . . 138
9.4 Theoretical Analysis . . . 139
9.5 Experiments . . . 142
9.5.1 Experiment Settings . . . 142
9.5.2 Results and Discussions . . . 145
9.6 Related Works . . . 146
9.7 Conclusion . . . 147
9.8 Limitations and Future Work . . . 147

10 Conformal Prediction for Language Models **148**
10.1 Introduction . . . 149
10.2 Preliminaries . . . 151
10.2.1 Multiple Choice Questions (MCQs) and LLMs . . . 151
10.2.2 Conformal Prediction . . . 153

10.3 Methodology . . . 154
10.3.1 Conformal Revision of Questions (CROQ) . . . 155
10.3.2 CP-OPT to Optimize Scores . . . 156
10.4 Experiments . . . 159
10.4.1 Experimental Setup . . . 160
10.4.2 Discussion . . . 162
10.5 Related Work . . . 164
10.6 Conclusions and Future Works . . . 165

A Appendix to Chapter 3 **195**
A.1 Definitions and Notation . . . 195
A.1.1 Basic Definitions . . . 195
A.1.2 Glossary . . . 198
A.2 Proofs . . . 198
A.2.1 Proofs for the General Setup . . . 198
A.2.2 Bounds for Finite VC-Dimension Classes . . . 208
A.2.3 Homogeneous Linear Classifiers with Uniform Distribution . . . 211
A.2.4 Lower Bound . . . 213
A.3 Additional Experiments . . . 214
A.3.1 Additional Experiments on Role of the Hypothesis Class . . . 214
A.3.2 Role of Confidence Function . . . 215
A.3.3 Detailed Results . . . 217
A.3.4 Auto Labeling Visualization . . . 218

B Appendix to Chapter 4 **228**
B.1 Appendix to the Background and Motivation Section . . . 228
B.1.1 Detailed comparison with active learning and self-training . . . 228
B.1.2 Details of the motivating experiment in section 4.2 . . . 229
B.2 Additional Details on the Method . . . 230
B.2.1 Detailed algorithms . . . 230
B.2.2 Tightness of surrogates. . . . 230
B.2.3 Active querying strategy. . . . 231
B.2.4 Glossary . . . 231
B.3 Additional Experiments and Details . . . 235
B.3.1 Train-time and post-hoc methods . . . 235
B.3.2 Detailed dataset and model . . . 236
B.3.3 Detailed experiments protocol . . . 237
B.3.4 Hyperparameters . . . 239
B.3.5 Compute resources and time . . . 240
B.3.6 Single vs multi-round TBAL . . . 240
B.3.7 Experiments on `Colander` input . . . 241
B.3.8 Experiments on ϵ_a . . . 242
B.3.9 Experiments on different architectures . . . 242
B.3.10 Experiments on N_t, N_v and ν . . . 243

C Appendix to Chapter 5 **247**

C.1 Glossary . . . 247
C.2 Detailed Algorithms . . . 247
C.3 Additional Experiments and Details . . . 248

D Appendix to Chapter 6 **256**
D.1 Glossary . . . 256
D.2 Proofs for Parameter Estimation Error in Discrete Spaces . . . 256
D.3 Proofs for Generalization Error in Discrete Space . . . 261
D.4 Proofs for Continuous Label Spaces . . . 271
D.5 Additional Details on Continuous Label Space . . . 279
D.6 Extended Background on Pseudo-Euclidean Embeddings . . . 280
D.6.1 Pseudo-metric Spaces . . . 281
D.6.2 Pseudo-Euclidean Spaces . . . 281

E Appendix to Chapter 8 **286**
E.1 Glossary . . . 286
E.1.1 Proofs . . . 286
E.1.2 Additional Details of the Algorithm . . . 297
E.2 Additional Experiments and Details . . . 297
E.2.1 Searching for Constants in LIL-Heuristic . . . 298
E.2.2 Additional Experiments on Synthetic Data . . . 298
E.2.3 Additional Real OOD Datasets Experiments . . . 299

F Appendix to Chapter 9 **309**
F.1 Glossary . . . 309
F.2 Additional Details for Methodology . . . 309
F.3 Proofs . . . 311
F.4 Additional Experiments and Details . . . 316
F.4.1 Additional Baseline . . . 316
F.4.2 Searching for Constants in Heuristic LIL UCB . . . 317
F.4.3 Choice of $\mathcal{G}$ . . . 318
F.4.4 Searching for Hyper-parameters . . . 318
F.4.5 Effects of Different Updating Frequencies . . . 319
F.4.6 Additional Experiments with Different Scoring Functions . . . 319
F.4.7 Additional Experiments on Different ID Datasets . . . 321

G Appendix to Chapter 10 **329**
G.1 Methodology and Background Details . . . 329
G.1.1 Details on LLM inference in multi-choice question answering . . . 329
G.2 Additional Experiments and Results . . . 330
G.2.1 Trade-off between coverage and accuracy . . . 330
G.2.2 Using conformal prediction for deferral . . . 331
G.2.3 MMLU . . . 333
G.2.4 TruthfulQA . . . 333
G.2.5 ToolAlpaca . . . 333
G.3 Calculation of Statistical Significance . . . 346

G.4 Example Questions and Prompts 346
G.4.1 MMLU 346
G.4.2 TruthfulQA 351
G.4.3 ToolAlpaca 353
G.5 Hyperparameter Settings 356

List of Figures

2.1 High-level diagram of TBAL system. 13
2.2 Illustration of outputs of labeling functions (observed) and groundtruth (latent) in weak supervision. Here 1 indicates positive and −1 negative labels, respectively. . 16
2.3 Weak supervision pipeline. 17

3.1 High-level workflow of threshold-based auto-labeling (TBAL). Box (B) shows the key component estimating the auto-labeling region using validation data and auto-labeling points in it. 21
3.2 Comparison of TBAL, active learning (AL) followed by selective classification (AL+SC) and passive learning (PL) followed by selective classification (PL+SC) on the Circles dataset (Sec. 3.2.3) using linear classifiers and confidence functions. (a) Samples auto-labeled, queried, and left unlabeled. (b) The auto-labeling error and coverage achieved by the algorithms. (50 trials.) 25
3.3 **Left:** Simplified upper bound from Corollary 3.4 (ignoring constants) on excess auto-labeling error for the Unit-Ball setting, i.e., homogeneous linear classifier with $d = 30$. **Right:** The worst observed auto-labeling error over 25 trials in the Unit-Ball experiment. 31
3.4 Results for varying N_q, the maximum number of samples the algorithm can use for training while providing sufficient validation samples. 36

4.1 High-level diagram of TBAL system. 41
4.2 Scores distributions (Kernel Density Estimates) of a CNN model trained on CIFAR-10 data. (a) softmax scores of the vanilla training procedure (SGD) (b) scores after post-hoc calibration using temperature scaling, and (c) scores from our `Colander` procedure applied on the same model. For training the CNN model, we use 4000 points drawn randomly and 1000 validation points (of which 500 are used for Temp. Scaling and `Colander`). The test accuracy of the model is 55%. Figures (d) and (e) show the coverage and auto-labeling error of these methods. The dotted-red line corresponds to a user-given error tolerance of 5%. 42

4.3 Threshold-based Auto-labeling with `Colander`: takes unlabeled data as input, selects a small subset of data points, and obtains human labels for them to create $D^{(i)}_{\text{train}}$ and $D^{(i)}_{\text{val}}$ for the ith iteration. Trains model $\hat{h}_i$ on $D^{(i)}_{\text{train}}$. In contrast to the standard TBAL procedure, here we randomly split $D^{(i)}_{\text{val}}$ into two parts, $D^{(i)}_{\text{cal}}$ and $D^{(i)}_{\text{th}}$. `Colander` kicks in, takes $\hat{h}_i$ and $D^{(i)}_{\text{cal}}$ as input and learns a coverage maximizing confidence function $\hat{g}_i$ for $\hat{h}_i$. Using $D^{(i)}_{\text{th}}$ and $\hat{g}_i$ auto-labeling thresholds $\hat{\mathbf{t}}_i$ are determined to ensure the auto-labeled data has error at most ϵ_a (a user-given parameter). After obtaining the thresholds, the rest of the steps are the same as standard TBAL. The whole workflow runs until all the data is labeled or another stopping criterion is met. 46
4.4 Our choice of g function. 49

5.1 Workflow of pseudolabeling-based SSL with `PabLO` (A): Train model with standard supervised loss, consistency regularization, and other regularizers (B): Learn new confidence scores and thresholds (C): Pseudolabel points with scores greater than the estimated thresholds (D): An optional pseudolabeling accumulation to use previous pseudolabels for points that are not pseudolabeled in the current round. Note that the pseudolabels can be noisy (incorrect). The training and pseudolabeling loop continues until a pre-specified number of iterations. In the end, it outputs a model $\widehat{h}_{\text{ssl}}$ that is expected to have higher test accuracy than the model trained only on the given groundtruth training data. Note that in the end, there might be points left unlabeled, and the pseudolabels might be noisy. 65
5.2 Top-1 accuracy of our method and baselines on CIFAR-10, CIFAR-100, and SVHN. We plot the values for every 200 steps. 70
5.3 Top-1 accuracy of our method with different error tolerance ϵ on (a) CIFAR-10 and (b) CIFAR-100 dataset. 73

6.1 Weak supervision pipeline. 81
6.2 Illustration of our weak supervision pipeline for the finite label space setting. . . 84
6.3 **Finite metric space case.** Parameter estimation improves with samples n in learning to rank—showing nearly-consistent behavior. Our tensor decomposition estimator outperforms (Shin et al., 2022a). In particular, (top left) as the number of samples increases, our estimates of the positive and negative components of $\mathbf{T}$ improve. (Top right) the improvements in $\mathbf{T}$ recovery with more samples translates to significantly improved performance over (Shin et al., 2022a), which is close to constant across n. (Bottom) this improved parameter estimation further translates to improvements in label model accuracy (using only the noisy estimates for prediction, without training an end model) and end model generalization. For the top two plots, we use $\boldsymbol{\theta} = [6, 3, 8]$, and in the bottom plot, we use $\boldsymbol{\theta} = [0, 0, 1]$. In all plots, we report medians along with upper and lower quartiles across 10 trials. 97
6.4 **Continuous case.** Parameter estimation improves with more samples in the hyperbolic regression problem. Our estimator outperforms Shin et al. (2022a). Here, we use different randomly sampled values of $\boldsymbol{\theta}$ for each run. We report medians along with upper and lower quartiles across 10 trials. 98

8.1 Illustration of OOD detection with human-in-the-loop with FPR control. In this example, the ID data is of brain scans of normal people and those with Alzheimer's disease. The OOD data could be anything other than these, e.g., brain scans of patients with some other diseases. 112
8.2 $\text{FPR}(\lambda^\star) = \alpha \longleftrightarrow \text{CDF}_{\mathcal{D}_{\text{out}}}(\lambda^\star) = 1-\alpha$. Optimal $\lambda^\star$ for the optimization problem (P1) with $\alpha = 0.05$ and $x_t \overset{i.i.d}{\sim} 0.7\,\mathcal{D}_{in} + 0.3\,\mathcal{D}_{out}$, where $\mathcal{D}_{in}$ is $\mathcal{N}(4,1)$ and $\mathcal{D}_{out}$ is $\mathcal{N}(0,1)$. 113
8.3 Illustration of the confidence interval defined in eq. (8.2) on FPR and their effect on threshold estimation. As the system receives more OOD samples, the confidence intervals will shrink and lead to better thresholds safely ($\hat{\lambda}_t \geq \lambda^\star$). 115
8.4 Results on the synthetic data with stationary distributions, $\gamma = 0.2$, and using no window. Each method is repeated 10 times. The mean and standard deviation are shown. . . . 120
8.5 Effect of using various window sizes in synthetic data experiments. The distribution shift starts at $t = 50k$. The arrow indicates the time at which the mean FPR + std. deviation over 10 runs goes below 5% for the LIL method. 120
8.6 Results with the KNN scores on CIFAR-10 as ID dataset. For (b) and (c) the distribution shifts at $t = 50k$. The arrow indicates the time at which the mean FPR + std. deviation over 10 runs goes below 5% for the LIL method. 121
8.7 Results on the synthetic data with stationary distributions and different mixing ratios γ. 123
8.8 Box plot of change detection times with different window sizes on synthetic data. The median and mean are shown using solid and dashed lines, respectively. 125

9.1 Comparison of score distributions between post-hoc fixed scoring functions (trained solely on ID data) and our learned function g^* (trained offline on both ID and OOD). CIFAR-10 and Places365 are the ID and OOD datasets, respectively. As t increases, the ASAT-adapted scoring functions converge toward g^*, which achieves the highest TPR at 5% FPR compared to EBO, KNN, and VIM methods from the Open-OOD benchmarks. This comparison suggests that incorporating human-labeled OOD inputs encountered during deployment enables ASAT to update scoring functions that better align with real-world OOD points. For more details, see Section 9.5. 133
9.3 Overview of the ASAT workflow, updating both scoring function g_t and thresholds λ_t via human feedback. 134
9.2 An example of OOD and ID score distributions mapped by g with a decision threshold λ. The hatched region is the FPR density. 134
9.4 Results on the stationary setting for CIFAR-10 as ID, with each method repeated 5 times (mean and std shown). EBO and KNN are the initial scoring functions and are fixed for FSFT and FSAT. The dotted-dash lines represent the *expected* TPR at FPR-5% for EBO, KNN, and $\widehat{g}^*$ (matching colors). 140
9.5 Results on the nonstationary setting, with each method repeated 5 times (mean and std shown). A shift happens between Near-OOD and Far-OOD mixtures at time $t = 50$k. EBO and KNN are the initial scoring functions and are fixed for FSFT and FSAT. The dotted-dash lines represent the TPR at FPR-5% for EBO, KNN, and $\widehat{g}^*$ (matching colors). 141
9.6 Results on the nonstationary setting with different window sizes 5k, 10k, and 15k, with each method repeated 5 times (mean and std shown). A shift happens from Places365 and CIFAR-100 at time $t = 50$k. 142

9.7 Results on nonstationary OOD from Places365 to CIFAR-100 with different window $N_w^{(est)}$; only ASAT results are shown. 142

10.1 Accuracy for three LLMs on the TruthfulQA dataset with 15 response options as a function of the number of incorrect answer options (distractors) removed from the prompt. As more distractor answers are eliminated, accuracy increases. Accuracy is averaged across 5 iterations, error bars denote ±2 standard deviations. 149

10.2 (CROQ) Illustration of conformal revision of questions and prompting the LLM with the revised question. In this example, the initial predicted set by LLM + conformal prediction (CP) is {C, D}. The question and labels are revised to contain only the answer choices in the prediction set and the LLM is prompted with the revised question. Since CP provides rigorous coverage guarantees, we expect that re-prompting the LLM with reduced answer choices will improve the chances of obtaining the correct answer. See Section 10.3.1 for more details. 155

A.1 Comparison of Threshold-Based Auto-Labeling (TBAL) and Active-Learning followed by Selective Classification (AL+SC) on XOR-dataset. Left figure (a) shows samples that were auto-labeled, queried, and left unlabeled by these methods. Figure (b) shows the auto-labeling error and coverage achieved. The lines show the mean, and the shaded region shows 1-standard deviation estimated over 10 trials with different random seeds. 214

A.2 Auto-labeling performance on MNIST data using different models (hypothesis classes) as a function of samples available for training. The left figure (a) shows the results with the linear classifier, and the right figure (b) shows the results with the LeNet classifier. The auto-labeling error threshold $\epsilon_a = 5\%$ in both experiments, and the algorithms are given the same amount of validation data. The lines show the mean, and the shaded region shows 1-standard deviation estimated over 5 trials with different random seeds. 216

A.3 Auto-labeling performance on CIFAR-10 data using a small network and different scoring functions. The left figure (a) shows the results with softmax scores, and the right figure (b) shows the results with the energy score. The auto-labeling error threshold $\epsilon_a = 10\%$ in both experiments and methods are given the same amount of validation data. The lines show the mean, and the shaded region shows 1-standard deviation estimated over 5 trials with different random seeds. 217

A.4 Histograms of scores computed on the validation data in a few rounds of TBAL run on CIFAR-10 with a small net. We picked two rounds where it auto-labeled the most, i.e., around 800 points. 217

A.5 Auto-labeling MNIST data using linear classifiers. Validation size = 12k. Maximum training samples = 1600. Each round algorithm queries 160 samples. Coverage of auto-labeling is 62.9% with 98.0% accuracy. For the rounds we show, the test error rates are 21.4%, 13.9%, 12.5%, 10.2%, and 9.8%, respectively. For four columns (left to right), we show: **a**) The samples that are labeled by TBAL in this round. **b**) The embeddings for training samples. **c**) The embeddings for validation data points. **d**) The score distribution for the validation dataset. 225

A.6 Auto-labeling CIFAR-10 data using a small network and energy scores. Validation size = 10k. Maximum training samples = 25k. Each round algorithm queries 2500 samples. Coverage of auto-labeling is 5.3% with 90.0% accuracy. For the rounds we show, the test error rates are 56.6%, 55.2%, 55.6%, 53.0%, and 49.3%, respectively. For four columns (left to right), we show: **a**) The samples that are labeled by TBAL in this round. **b**) The embeddings for training samples. **c**) The embeddings for validation data points. **d**) The score distribution for the validation dataset. 226

A.7 Auto-labeling CIFAR-10 data using a small network and softmax scores. Validation size = 10k. Maximum training samples = 25k. Each round of the algorithm queries 2500 samples. Coverage of auto-labeling is 2.3% with 91.0% accuracy. For the rounds visualized here in each row, the test error rates of the trained classifiers are 56.6%, 59.1%, 52.8%, 50.5%, and 51.7% respectively. For four columns (left to right), we show: **a**) The samples that are labeled by TBAL in this round. **b**) The embeddings for training samples. **c**) The embeddings for validation data points. **d**) The score distribution for the validation dataset. 227

B.1 Results of experiment on 2-concentric circles to show the differences between TBAL, AL, and ST. 229

B.2 Illustration of the tightness of surrogate error and coverage functions based on the choice of α. 230

B.3 Per-epoch metrics for all post-hoc methods for CIFAR10 setting. (left) Auto-labeling accuracy (right) Coverage. Train-time method is vanilla. 240

B.4 Autolabeling error and coverage of different post-hoc methods on CIFAR-10 for various N_t 244

B.5 Autolabeling error and coverage of different post-hoc methods on CIFAR-10 for various N_v 244

B.6 Autolabeling error and coverage of different post-hoc methods on CIFAR-10 for various ν 244

B.7 Auto-labeling error and coverage for different post-hoc methods on CIFAR-10 while we vary N_t. $N_u = 40,000$ is the size of the given unlabeled pool. 245

B.8 Auto-labeling error and coverage for different post-hoc methods on Tiny-ImageNet while we vary N_t. $N_u = 90,000$ is the size of the given unlabeled pool. 245

B.9 Auto-labeling error and coverage for different post-hoc methods on 20 Newsgroups while we vary N_t. $N_u = 9,052$ is the size of the given unlabeled pool. 245

B.10 Auto-labeling error and coverage for different post-hoc methods on CIFAR-10 while we vary N_v. $N_{v_{\max}} = 8,000$ is the maximum number of points available for validation. 246

B.11 Auto-labeling error and coverage for different post-hoc methods on Tiny-ImageNet while we vary N_v. $N_{v_{\max}} = 18,000$ is the maximum number of points available for validation. 246

B.12 Auto-labeling error and coverage for different post-hoc methods on 20 Newsgroups while we vary N_v. $N_{v_{\max}} = 1,600$ is the maximum number of points available for validation. 246

C.1 Batch pseudolabel accuracy of our method and baselines on CIFAR-10, CIFAR-100, and SVHN. We plot the values for every 200 steps. 250
C.2 Batch pseudolabel coverage of our method and baselines on CIFAR-10, CIFAR-100, and SVHN. We plot the values for every 200 steps. 251
C.3 **(A1.)** Left to Right: Top-1 Accuracy, Batched pseudolabeling Accuracy, and batched pseudolabeling coverage of Fixmatch with and without pseudolabeling accumulation enabled on CIFAR-10. It can be seen that enabling pseudolabeling accumulation worsens the performance of baseline methods in terms of accuracy and coverage. 252
C.4 **(A1.)** Left to Right: Top-1 Accuracy, Batched pseudolabeling Accuracy, and batched pseudolabeling coverage of Freematch with and without pseudolabeling accumulation enabled on CIFAR-10. It can be seen that enabling pseudolabeling accumulation worsens the performance of baseline methods in terms of accuracy and coverage. 252
C.5 Top-1 accuracy of our method with different N_{th} and N_{cal}. 254

E.1 Changing ID distribution in synthetic data. 298
E.2 Results with the KNN scores on CIFAR-10 as ID dataset. For (b) and (c) the distribution shifts at $t = 50k$. The arrow indicates the time at which the mean FPR + std. deviation over 10 runs goes below 5% for the LIL method. 302
E.3 Results with the EBO scores on CIFAR-10 as ID dataset. For (b) and (c) the distribution shifts at $t = 50k$. The arrow indicates the time at which the mean FPR + std. deviation over 10 runs goes below 5% for the LIL method. 302
E.4 Results with the MDS scores on CIFAR-10 as ID dataset. For (b) and (c) the distribution shifts at $t = 50k$. The arrow indicates the time at which the mean FPR + std. deviation over 10 runs goes below 5% for the LIL method. 302
E.5 Results with the VIM scores on CIFAR-10 as ID dataset. For (b) and (c) the distribution shifts at $t = 50k$. The arrow indicates the time at which the mean FPR + std. deviation over 10 runs goes below 5% for the LIL method. 303
E.6 Results with the ODIN scores on CIFAR-10 as ID dataset. For (b) and (c) the distribution shifts at $t = 50k$. The arrow indicates the time at which the mean FPR + std. deviation over 10 runs goes below 5% for the LIL method. 303
E.7 Results with the SSD scores on CIFAR-10 as ID dataset. For (b) and (c) the distribution shifts at $t = 50k$. The arrow indicates the time at which the mean FPR + std. deviation over 10 runs goes below 5% for the LIL method. 303
E.8 Results with the KNN scores on CIFAR-100 as ID dataset. For (b) and (c) the distribution shifts at $t = 50k$. The arrow indicates the time at which the mean FPR + std. deviation over 10 runs goes below 5% for the LIL method. 304
E.9 Results with the EBO scores on CIFAR-100 as ID dataset. For (b) and (c) the distribution shifts at $t = 50k$. The arrow indicates the time at which the mean FPR + std. deviation over 10 runs goes below 5% for the LIL method. 304
E.10 Results with the MDS scores on CIFAR-100 as ID dataset. For (b) and (c) the distribution shifts at $t = 50k$. The arrow indicates the time at which the mean FPR + std. deviation over 10 runs goes below 5% for the LIL method. 304

E.11 Results with the VIM scores on CIFAR-100 as ID dataset. For (b) and (c) the distribution shifts at $t = 50k$. The arrow indicates the time at which the mean FPR + std. deviation over 10 runs goes below 5% for the LIL method. 305
E.12 Results with the SSD scores on CIFAR-100 as ID dataset. For (b) and (c) the distribution shifts at $t = 50k$. The arrow indicates the time at which the mean FPR + std. deviation over 10 runs goes below 5% for the LIL method. 305
E.13 Results with the ODIN scores on CIFAR-100 as ID dataset. For (b) and (c) the distribution shifts at $t = 50k$. The arrow indicates the time at which the mean FPR + std. deviation over 10 runs goes below 5% for the LIL method. 305
E.14 Scores distribution for KNN with CIFAR-10 as In-Distribution. 306
E.15 Scores distribution for EBO with CIFAR-10 as In-Distribution. 306
E.16 Scores distribution for SSD with CIFAR-10 as In-Distribution. 306
E.17 Scores distribution for VIM with CIFAR-10 as In-Distribution. 307
E.18 Scores distribution for MDS with CIFAR-10 as In-Distribution. 307
E.19 Scores distribution for ODIN with CIFAR-10 as In-Distribution. 307
E.20 Scores distribution for KNN with cifar-100 as In-Distribution. 307
E.21 Scores distribution for EBO with Cifar-100 as In-Distribution. 307
E.22 Scores distribution for SSD with Cifar-100 as In-Distribution. 308
E.23 Scores distribution for VIM with Cifar-100 as In-Distribution. 308
E.24 Scores distribution for MDS with Cifar-100 as In-Distribution. 308
E.25 Scores distribution for ODIN with Cifar-100 as In-Distribution. 308

F.1 Comparison between ASAT and a naive binary classifier baseline. ASAT is initialized with KNN. CIFAR-10 as ID and CIFAR-100 and Near-OOD as OOD. Each method is repeated five times and is shown with mean and std. deviation. The binary classifier is trained using cross-entropy loss on the ID and OOD data collected up to time t, without importance sampling (since the classification is not based on score thresholding). . . . 317
F.2 Results on the stationary setting with different update frequencies, with each method repeated 5 times (mean and std shown). CIFAR-10 as ID and Places365 as OOD. Optimization frequencies are set uniformly for 100, 500, 1000, and 2500 new OOD samples. We fix $c_1 = 0.5$ in the heuristic UCB $\psi_t^H(\delta)$. 320
F.3 Results for stationary settings using CIFAR-10 as ID, with each method repeated 5 times (mean and std shown). Run with five additional scoring functions, MDS, VIM, ODIN, ASH, and SHE. 325
F.4 Results for nonstationary OOD settings using CIFAR-10 as ID, with each method repeated 5 times (mean and std shown). OOD distribution shifts from Far-OOD to Near-OOD. Run with four additional scoring functions, MDS, VIM, ODIN, and SHE. 326
F.5 Results for stationary settings using CIFAR-100 as ID, with each method repeated 5 times (mean and std shown). Run with four scoring functions, EBO, KNN, VIM, and ODIN. 327
F.6 Results for nonstationary settings using CIFAR-100 as ID, with each method repeated 5 times (mean and std shown). OOD distribution shifts from Far-OOD to Near-OOD. Run with four additional scoring functions, MDS, VIM, ODIN, and SHE. 328
F.7 Results on the stationary setting for ImageNet-1k as ID, with each method repeated 5 times (mean and std shown). KNN and MDS are the initial scoring functions and are fixed for FSFT and FSAT. The dotted-dash lines represent the *expected* TPR at FPR-5% for KNN, MDS, and $\widehat{g}^*$ (matching colors). 328

G.1 Proportion of questions deferred to a human when conformal prediction set sizes exceed a certain cutoff (left), and the corresponding LLM accuracy for questions (without revision) retained by the LLM as a function of cutoff threshold (right). In the top row (MMLU, 10 options, `Phi-3-4k-mini-Instruct`), the difference in deferral and accuracy is negligible, whereas in the bottom row (TruthfulQA, 15 options, `gemma-2-9b-it-SimPO`), CP-OPT defers fewer questions to the human while providing similar or improved accuracy for questions retained. 332
G.2 Accuracy on revised questions on the MMLU and ToolAlpaca datasets while varying miscoverage parameter α for `Phi-3-4k-mini-Instruct` (Phi-3) model and both scores. Smaller values of α correspond to high levels of coverage. When coverage is too large, few or no answers are eliminated, and the LLM is prompted with the same question. When coverage is low, a larger portion of answer sets no longer contain the true answer, and the benefits of revision are diminished. 333
G.3 Accuracy on revised questions on the MMLU and ToolAlpaca datasets while varying miscoverage parameter α for `Llama-3-8B-Instruct` (Llama-3) model and both scores. Smaller values of α correspond to high levels of coverage. When coverage is too large, few or no answers are eliminated, and the LLM is prompted with the same question. When coverage is low, a larger portion of answer sets no longer contain the true answer or produce empty prediction sets, thus resulting in diminished benefits of revision. 334
G.4 Distributions of sizes of sets obtained from CP-OPT and logit scores on MMLU dataset and Gemma-2 model. 334
G.5 Distributions of sizes of sets obtained from CP-OPT and logit scores on MMLU dataset and Llama-3 model. 335
G.6 Distributions of sizes of sets obtained from CP-OPT and logit scores on MMLU dataset and Phi-3 model setting. 335
G.7 Distributions of sizes of sets obtained from CP-OPT and logit scores on ToolAlpaca dataset and Gemma-2 model. 335
G.8 Distributions of sizes of sets obtained from CP-OPT and logit scores on ToolAlpaca dataset and Llama-3 model. 336
G.9 Distributions of sizes of sets obtained from CP-OPT and logit scores on ToolAlpaca dataset and Phi-3 model. 336
G.10 Distributions of sizes of sets obtained from CP-OPT and logit scores on Truthful QA dataset and Gemma-2. 336
G.11 Distributions of sizes of sets obtained from CP-OPT and logit scores on Truthful QA dataset and Phi-3 model. 337
G.12 Distributions of sizes of sets obtained from CP-OPT and logit scores on Truthful QA dataset and Llama-3 model. 337

List of Tables

3.1 **Unit-Ball.** Effect of variation of validation data size (N_v) with and without using a UCB on error estimates. We keep training data size N_q fixed at 500 and use error threshold $\epsilon_a = 1\%$. We report the mean and std. deviation over 10 runs with different random seeds. **Left**: with $C_1 = 0$. **Right**: with $C_1 = 0.25$. 33

3.2 **IMDB.** Effect of variation of validation data size (N_v) with and without using a UCB on error estimates. We keep training data size N_q fixed at 500 and use error threshold $\epsilon_a = 5\%$. We report the mean and std. deviation over 10 runs with different random seeds. **Left:** with $C_1 = 0$. **Right:** with $C_1 = 0.25$. 33

3.3 **Tiny-ImageNet.** Effect of variation of validation data size (N_v) with and without using a UCB on error estimates. We keep the training data size N_q fixed at $10k$ and use an error threshold $\epsilon_a = 10\%$. We report the mean and std. deviation over 5 runs with different random seeds. **Left:** with $C_1 = 0$. **Right:** with $C_1 = 0.25$. . 34

3.4 Results for varying N_q, the maximum number of samples the algorithm can use for training. **Left: IMDB** with $N_v = 1000, C_1 = 0.25, \epsilon_a = 5\%$. **Right: Tiny-ImageNet** with $N_v = 10K, C_1 = 0.25, \epsilon_a = 10\%$. Mean and std. deviations are reported. 35

4.1 Details of the dataset and model we used to evaluate the performance of our method and other calibration methods. For the Tiny-Imagenet and 20 Newsgroup datasets, we use CLIP and FlagEmbedding, respectively, to obtain the embeddings of these datasets and conduct auto-labeling on the embedding space. For Tiny-Imagenet, we use a 3-layer perceptron with 1,000, 500, 300 neurons on each layer as model $\hbar$; for 20 Newsgroup, we use a 3-layer perceptron with 1,000, 500, 30 neurons on each layer as model $\hbar$. 52

4.2 In every round, the error was enforced to be below 5%; 'TS' stands for Temperature Scaling, 'SB' stands for Scaling Binning, 'Top-HB' stands for Top-Label Histogram Binning. 'AdaTS' stands for Adaptive Temperature Scaling. The column Err stands for auto-labeling error, and Cov stands for coverage. Each cell value is the mean ± std. deviation on 5 repeated runs with different random seeds. 58

5.1 Details of the dataset we use in our experiments. k is the number of classes. N_l is the number of labeled data points used for training the backbone model h. n_u is the number of unlabelled data points used for consistency regularization and pseudolabeling for all the methods. N_{val} is the number of points used for model selection in all methods. N_{test} is the number of test data points. N_{cal} is the number of points used for learning the g function. N_{th} is the number of data points used for threshold estimation. . . . 67
5.2 Top-1 Accuracy for CIFAR-10, CIFAR-100 and SVHN averaged across 3 random seeds. The best accuracy is **bolded.** . . . 72
5.3 Top-1 accuracy for the two error tolerance (ϵ) scheduling. The table reports the ϵ we use between each iteration interval and the top-1 accuracy yielded by the corresponding schedule. For schedule 1, we set $\epsilon = 0.4, 0.2, 0.1, 0.05, 0.01$ when the number of iterations is in the following interval, respectively: $[0, 5\text{K})$, $[5\text{K}, 10\text{K})$, $[10\text{K}, 15\text{K})$, $[15\text{K}, 20\text{K})$, $[20\text{K}, 25\text{K})$. For schedule 2, we set $\epsilon = 0.01, 0.05, 0.1, 0.2, 0.4$, for the same intervals, respectively. . . . 74
5.4 Results on CIFAR-10 with and without pseudolabel accumulation (Acc) for all the methods. . . . 75

8.1 Time to reach feasibility T_f and optimality $T_{\eta-\text{opt}}$ in the stationary setting for different η and mixing ratios γ. . . . 124

9.1 Comparison of scoring functions, thresholds, FPR control, and TPR for FSFT, FSAT, and ASAT. . . . 132

10.1 Average set sizes and coverage rates (in percentages) for conformal prediction sets on the MMLU, ToolAlpaca, and TruthfulQA datasets using `gemma-2-9b-it-SimPO` (Gemma-2), `Llama-3-8B-Instruct` (Llama-3) and `Phi-3-4k-mini-Instruct` (Phi-3), with a target coverage level of 95%. Bold numbers indicate smaller average set sizes. Asterisks on the larger of a pair of numbers indicate where the difference in average set size or coverage is statistically significant at the 0.05 significance level. 160
10.2 [CROQ + logits]. Results on accuracy improvement with CROQ using logit scores. Here a_1, and a_1' refer to the accuracy before CROQ and after CROQ respectively. A positive gain implies CROQ improved accuracy in that setting. . . . 161
10.3 [CROQ + logits vs CROQ + CP-OPT]. Comparison of CP-OPT and logits on accuracy improvement with CROQ. Here, a_1', and a_2' are the final accuracies after CROQ using logits and CP-OPT respectively (as in Tables 10.2 and G.1. The gain $a_2' - a_1'$ is the difference between these two, with values indicating more improvement in CROQ with CP-OPT scores. . . . 161

A.1 Glossary of variables and symbols used in this paper. . . . 220
A.2 **IMDB**. Effect of variation of validation data size (N_v) without using a UCB (i.e., $C_1 = 0$) on error estimates. We keep training data size N_q fixed at 500 and use error threshold $\epsilon_a = 5\%$. We report the mean and std. deviation over 10 runs with different random seeds. . . . 221

A.3 **IMDB**. Effect of variation of validation data size (N_v), using a UCB (i.e., $C_1 = 0.25$) on error estimates. We keep training data size N_q fixed at 500 and use error threshold $\epsilon_a = 5\%$. We report the mean and std. deviation over 10 runs with different random seeds. 221
A.4 **IMDB**. Effect of variation of N_q, the maximum number of samples the algorithm can use for training, without using a UCB (i.e., $C_1 = 0$) on error estimates. We keep validation data size N_v fixed at 1000 and use error threshold $\epsilon_a = 5\%$. We report the mean and std. deviation over 10 runs with different random seeds. . . 221
A.5 **IMDB**. Effect of variation of N_q, the maximum number of samples the algorithm can use for training, using a UCB (i.e., $C_1 = 0.25$) on error estimates. We keep validation data size N_v fixed at 1000 and use error threshold $\epsilon_a = 5\%$. We report the mean and std. deviation over 10 runs with different random seeds. 222
A.6 **Unit Ball**. Effect of variation of validation data size (N_v), without using a UCB (i.e., $C_1 = 0$) on error estimates. We keep training data size N_q fixed at 500 and use error threshold $\epsilon_a = 1\%$. We report the mean and std. deviation over 10 runs with different random seeds. 222
A.7 **Unit-Ball**. Effect of variation of validation data size (N_v), without using a UCB (i.e., $C_1 = 0.25$) on error estimates. We keep training data size N_q fixed at 500 and use error threshold $\epsilon_a = 1\%$. We report the mean and std. deviation over 10 runs with different random seeds. 222
A.8 **Unit-Ball**. Effect of variation of N_q, the maximum number of samples the algorithm can use for training, without using a UCB (i.e., $C_1 = 0$) on error estimates. We keep validation data size N_v fixed at 4000 and use error threshold $\epsilon_a = 1\%$. We report the mean and std. deviation over 10 runs with different random seeds. 223
A.9 **Unit-Ball**. Effect of variation of N_q, the maximum number of samples the algorithm can use for training, using a UCB (i.e., $C_1 = 0.25$) on error estimates. We keep validation data size N_v fixed at 4000 and use error threshold $\epsilon_a = 1\%$. We report the mean and std. deviation over 10 runs with different random seeds. 223
A.10 **Tiny-ImageNet**. Effect of variation of validation data size (N_v), without using a UCB (i.e., $C_1 = 0$) on error estimates. We keep training data size N_q fixed at $10K$ and use error threshold $\epsilon_a = 10\%$. We report the mean and std. deviation over 10 runs with different random seeds. 223
A.11 **Tiny-ImageNet**. Effect of variation of validation data size (N_v), using a UCB (i.e., $C_1 = 0.25$) on error estimates. We keep training data size N_q fixed at $10K$ and use error threshold $\epsilon_a = 10\%$. We report the mean and std. deviation over 10 runs with different random seeds. 224
A.12 **Tiny-ImageNet**. Effect of variation of N_q, the maximum number of samples the algorithm can use for training, without using a UCB (i.e., $C_1 = 0$) on error estimates. We keep validation data size N_v fixed at $10K$ and use error threshold $\epsilon_a = 10\%$. We report the mean and std. deviation over 5 runs with different random seeds. 224

A.13 **Tiny-ImageNet**. Effect of variation of N_q, the maximum number of samples the algorithm can use for training, using a UCB (i.e., $C_1 = 0.25$) on error estimates. We keep validation data size N_v fixed at $10K$ and use error threshold $\epsilon_a = 10\%$. We report the mean and std. deviation over 5 runs with different random seeds. . 224

B.1 Glossary of variables and symbols used in this paper. 234
B.2 Hyperparameters swept over for train-time methods. Those listed next to Common are the hyperparameters for the four train-time methods: Vanilla, CRL, FMFP, and Squentropy. Therefore, we do not list those again for each method. Note that for FMFP, we used SAM optimizer instead of SGD. For each method, we swept through all possible combinations of the possible values for each hyperparameter. Underlined values are only used on TinyImageNet since it is a complicated dataset containing 200 classes. 237
B.3 Hyperparameters swept over for post-hoc methods. For each method, we swept through all possible combinations of the possible values for each hyperparameter. 238
B.4 Results with a single round of auto-labeling. Dataset and model: CIFAR-10 setting in the paper. 240
B.5 Auto-labeling error and coverage for the 3 feature representations we could use for 20 Newsgroup. As we can see, the feature representation does not lead to a significant difference in auto-labeling error and coverage. 241
B.6 Auto-labeling error and coverage for the 3 feature representations we could use for CIFAR10 SimpleCNN. As we can see, the feature representation does not lead to a significant difference in auto-labeling error and coverage. 241
B.7 ϵ_a **variation.** Dataset: CIFAR-10, Train-time method: Vanilla. 242
B.8 Model variation. CIFAR-10 dataset with ViT (Left) and ResNet18 (Right), Train-time method Vanilla. 242

C.1 Glossary of variables and symbols used in this paper. 253
C.2 Results on CIFAR-10 with varying N_{cal} and N_{th}. 254
C.3 Results on CIFAR-10 with varying ϵ. 254
C.4 Results on CIFAR-100 with varying ϵ. 254
C.5 Hyperparameters used for our method. 255

D.1 Glossary of variables and symbols used in this paper. 285

E.1 Glossary of variables and symbols used in this paper. 287

F.1 Glossary of variables and symbols used in the framework. 323
F.2 Lists of hyper-parameters we swept over to find the best combination for training $g \in \mathcal{G}_1$ and $\lambda \in \Lambda$ in optimization (P9.2). 324

G.1 [CROQ + CP-OPT]. Results on accuracy improvement with CROQ using CP-OPT scores. Here, a_2, and a_2' refer to the accuracy before CROQ and after CROQ, respectively. A higher gain in a setting suggests CROQ improved accuracy in that setting. 337
G.2 Results for CROQ on the MMLU dataset with 15 response options and Gemma-2 model. 338

G.3 Results for CROQ on the MMLU dataset with 15 response options and Llama-3 model. 338
G.4 Results for CROQ on the MMLU dataset with 15 response options and Phi-3 model. 338
G.5 Results for CROQ on the MMLU dataset with 4 response options. 339
G.6 Results for CROQ on the MMLU dataset with 10 response options. 340
G.7 Results for CROQ on the TruthfulQA dataset with 15 response options and Gemma-2 model 340
G.8 Results for CROQ on the TruthfulQA dataset with 15 response options and Llama-3. 341
G.9 Results for CROQ on the TruthfulQA dataset with 15 response options and Phi-3 model. 341
G.10 Results for CROQ on the TruthfulQA dataset with 10 response options. 341
G.11 Results for CROQ on the TruthfulQA dataset with 4 response options. 342
G.12 Results for CROQ on the ToolAlpaca dataset with 4 response options. 343
G.13 Results for CROQ on the ToolAlpaca dataset with 10 response options. 344
G.14 Results for CROQ on the ToolAlpaca dataset with 15 response options and Gemma-2. 344
G.15 Results for CROQ on the ToolAlpaca dataset with 15 response options and Llama-3 model. 345
G.16 Results for CROQ on the ToolAlpaca dataset with 15 response options and Phi-3 model. 345
G.17 Hyperparameter settings for our score function learning procedure CP-OPT in our experiments. For all settings we use SGD with momentum 0.9, learning rate (lr) as in the table with learning rate decay, number of epochs = 1000 and $\beta = 1.0$. . 356

Chapter 1

Organization

This thesis is divided into two parts. The first part, "Tackling the Data Bottleneck" (Part I), focuses on auto-labeling solutions to alleviate the data bottleneck. This part is structured as follows,

- Chapter 2 motivates the problem of the labeled data bottleneck and provides a concise background on auto-labeling solutions, including threshold-based auto-labeling and weak supervision. It summarizes our contributions toward advancing these techniques and situates them within the broader literature on auto-labeling and learning with limited supervision. Finally, it offers an outlook for future research directions and outlines key open questions.
- Chapter 3 is based on "Promises and Pitfalls of Threshold-based Auto-labeling" (Vishwakarma et al., 2023). It provides the foundations and theoretical analysis of TBAL.
- Chapter 4 is based on "Pearls from Pebbles: Improved Confidence Functions for Auto-labeling" (Vishwakarma et al., 2024a). This chapter dives deeper into the challenges in TBAL arising from the choice of confidence scores and presents a solution (Colander) to mitigate these challenges.
- Chapter 5 focuses on pseudolabeling-based semi-supervised learning and proposes improvements to it. This chapter is based on "PabLO: Improving Semi-Supervised Learning with

Pseudolabeling Optimization" (Vishwakarma et al., 2024b).

- Chapter 6 is based on "Lifting Weak Supervision To Structured Prediction" (Vishwakarma and Sala, 2022). Here, we present our approach to extend weak supervision (WS) to structured label spaces and derive generalization error bounds while learning with labels from our WS procedure.

The second part of the thesis, "Improving Reliability and Accuracy at Inference-time" (Part II) is centered on techniques to improve the reliability and accuracy of predictions of the models at test time. This part is organized into the following chapters,

- Chapter 7 provides an overview of this part. It motivates the broader challenges of achieving test-time reliability and accuracy, and highlights specific problems such as robustness to out-of-distribution (OOD) inputs and improving large language model (LLM) performance at inference. The chapter introduces key techniques used in this part, including conformal prediction and anytime valid inference, summarizes our contributions in the context of related literature, and outlines promising directions for future work.

- Chapter 8 is based on "Taming False Positives in Out-of-Distribution Detection with Human Feedback" (Vishwakarma et al., 2024c). This chapter introduces a human-in-the-loop framework that adapts OOD-detection thresholds in response to OOD inputs, while maintaining strict control on the false positive rate (FPR) at all times.

- Chapter 9 is based on "Adaptive Scoring and Thresholding with Human Feedback for Robust Out-of-Distribution Detection" (Yamada et al., 2025). It builds on Chapter 8 and incorporates a principled method to adapt scoring functions over time along with the thresholds, to maximize the true positive rate (TPR) while controlling FPR at all times.

- Chapter 10 is based on "Monty Hall and Optimized Conformal Prediction to Improve Decision-Making with LLMs" (Vishwakarma et al., 2024d). This chapter explores ways to improve LLM inference in structured settings using conformal prediction.

In addition, the author has also contributed to research on other topics that are not part of this

thesis. Some of them include robustness to backdoor attacks in federated learning (Wang et al., 2020a), theoretical analysis of lottery ticket hypothesis (Pensia et al., 2020), parameter markets (Huang et al., 2023a), automating programmatic weak supervision (Huang et al., 2023b) and zero-shot adaptation of foundation models (Shin et al., 2024a).

Part I

Tackling the Data Bottleneck

Chapter 2

Overview

2.1 Motivation

One major obstacle in developing ML solutions is the *labeled data bottleneck*, resulting from the following two factors.

(i) The demand for labeled data in ML is perpetual. The idea to *learn from data* marked a significant departure in developing intelligent applications. In contrast to hand-crafting rules, it allowed the system to learn the "rules" and patterns directly from the data. This approach provides a lot of flexibility and has become the predominant choice for developing intelligent systems. But the success now hinges upon the availability of high-quality labeled data — even in the era of large-scale pre-trained foundation models as outlined below,

1. *Novel applications.* The range of ML applications keeps expanding, ranging from classical problems of spam classification to detecting novel diseases. In the majority of such use cases, new high-quality labeled data is essential to success.

2. *Fine-tuning and alignment with human preferences and values.* Large-scale pretraining has proved to be highly successful in utilizing the vast amount of unlabeled, noisy Internet data. However, the models learned from such a process require task-specific adaptation (or fine-tuning) to be effective at any task of interest, e.g., machine translation. Such

fine-tuning procedures rely on a large amount of high-quality annotated data.

3. *Model is as good as the quality and quantity of data.* Lastly, statistical learning theory and practical experiences suggest that, with more data, a model can always be improved, as it can help in capturing more unseen (under-covered) parts of the space.

(ii) Challenges in obtaining high-quality labeled data. On one hand, we see the ever-increasing demand for labeled data in machine learning, and on the other hand, there are obstacles on the supply side. Some of them are,

1. *Expensive and time-consuming process.* Usual solution to acquire such data is to collect sufficiently large unlabeled samples and then, depending on the complexity of the task, either hire some expert(s) to label them or use crowdsourcing platforms such as Amazon Mechanical Turk to obtain labels. This process can be highly expensive and time-consuming, even for classification tasks with fewer categories.

2. *Harder with structured labels.* The problem is more challenging while dealing with *diverse tasks* such as when labels are in general metric spaces or manifolds. For example, collecting chain-of-thought (CoT) trees for reasoning problems, dependency parse trees to parse natural language sentences, rankings of user preferences, etc. Such settings make the annotation tasks significantly more demanding, resulting in much higher costs.

There are several solutions used in practice to either learn from fewer labels or create labeled datasets at low cost by leveraging ML models and other cheaply available sources. In this thesis, we focus on understanding and enhancing two popular commercial techniques for auto-labeling – **weak supervision** (Snorkel, 2021) and **threshold-based auto-labeling** (SGT, 2022), and study **self-training** approaches to learn with less labels. Our research is guided by two questions,

Q1. Conditions for efficient and accurate auto-labeling. Given the long shelf-life and diverse usage of the resulting datasets, the accuracy of labeled data is highly important. Which motivates the question, what are the conditions under which these systems can ensure a certain level of accuracy on the auto-labeled data?

Q2. Generalization to diverse tasks. How can we extend the auto-labeling solutions to applications where labels are more complex, e.g., CoT, parse trees, etc?

2.2 Literature

The data bottleneck in machine learning has led to a wide range of solutions over the decades. Crowdsourcing, where human annotators provide labels at scale—often with strategies to manage noisy or inconsistent inputs (Dekel and Shamir, 2009; Raykar et al., 2010; Vinayak et al., 2014; Chen et al., 2023a)—remains one of the most widely used approaches for obtaining labeled data. In parallel, various learning paradigms have been developed to reduce reliance on manual labeling, including semi-supervised learning (Board and Pitt, 1989; Zhu, 2005; Amini et al., 2023), active learning (Cohn et al., 1994; Balcan et al., 2007; Zhu and Nowak, 2022), transfer learning (Caruana, 1997; Torrey and Shavlik, 2010; Zhuang et al., 2020), and learning latent variable models (Bishop, 1998).

The focus has predominantly been on *learning* good models with less data, and crowdsourcing (or expert labeling) has been the primary means to obtain labeled data (Deng et al., 2009). There have been some sporadic efforts on improving the data labeling processes, for instance, using model generated labels to guide human annotators or utilizing domain specific insights in concert with the aforementioned techniques to label data (Gildea and Jurafsky, 2002; Desikan et al., 2006; Mehrotra et al., 2013; Shi et al., 2018; Zhang et al., 2021a; Asano et al., 2020; Desmond et al., 2021). More recently, two commercial techniques for creating labeled datasets have become widely used: Threshold-based auto-labeling (TBAL) (SGT, 2022; Qiu et al., 2023) and Weak Supervision (WS) (Ratner et al., 2016; Varma et al., 2019; Fu et al., 2020). This thesis focuses on providing a theoretical understanding and improving these two techniques.

TBAL is a recent technique to create labeled datasets using human labels and model-predicted labels (auto-labels). It draws on ideas from pseudolabeling-based semi-supervised learning (Lee et al., 2013), active learning, and selective classification (El-Yaniv and Wiener, 2010). While TBAL has seen commercial success, a fundamental theoretical understanding has been lacking.

This thesis presents the first rigorous foundation for TBAL and proposes improvements to the method. WS techniques build upon the developments in learning graphical models and latent variable models (Wainwright and Jordan, 2008; Bishop, 1994) to create labeled datasets by aggregating noisy labels from multiple sources.

Semi-supervised Learning (SSL). It aims to improve learning using both labeled and unlabeled data (Zhu, 2005). Classical methods include generative models (Nigam et al., 2000), graph-based label propagation (Zhu et al., 2003; Song et al., 2021), and low-density separation (Joachims, 1999). Recent deep learning approaches rely on pseudo-labeling (Lee et al., 2013) and consistency regularization (Sohn et al., 2020). TBAL combines ideas from pseudo-labeling, selective prediction and active learning.

Selective Classification. It allows a classifier to abstain from making predictions to ensure high accuracy. Foundational works (El-Yaniv and Wiener, 2010; Wiener and El-Yaniv, 2011; El-Yaniv and Wiener, 2012; Wiener and El-Yaniv, 2015) provide theoretical guarantees on error and coverage but offer limited practical algorithms. Later work (Gelbhart and El-Yaniv, 2019) introduces a disagreement-based active learning approach, while (Geifman and El-Yaniv, 2017) proposes a threshold-based method for deep networks with error bounds derived from (Gascuel and Caraux, 1992). However, these assume a fixed classifier. In contrast, TBAL learns classifiers dynamically via ERM on adaptively sampled data, applying SC at each round rather than once on a fixed model.

Active Learning. It aims to reduce labeling effort by selecting the most informative examples (Settles, 2009; Hanneke, 2014; Ren et al., 2020). While it works well under low noise and correct model assumptions, real-world settings may violate these conditions. In such cases, its label efficiency can match that of passive learning (Kääriäinen, 2006).n contrast, TBAL does not rely on idealized assumptions—our work shows it remains effective even under model misspecification.

Learning Latent Variable Models. Latent variable models are essential tools for capturing hidden structures in data (Bishop, 1998; Anandkumar et al., 2014a). The expectation-maximization (EM) algorithm is widely used for training these models, enabling them to uncover latent variables

that explain observed patterns. Work by Dawid and Skene (1979) has applied latent variable models to crowdsourcing, modeling the uncertainty in human-provided labels. Weak supervision techniques, such as data programming (Ratner et al., 2016; Varma et al., 2019; Fu et al., 2020), also rely on latent variable models to infer groundtruth labels by aggregating noisy labels from multiple weak sources. In this thesis, we extend these approaches to settings where structured label spaces, e.g., rankings, trees, etc.

2.3 Contributions

In this section, we provide a summary of our contributions towards tackling the data bottleneck in machine learning. Specifically, we study and improve threshold-based auto-labeling (TBAL) (Chapters 3, 4), pseudolabeling-based semi-supervised learning in Chapter 5, and lift weak supervision to settings with structured label spaces (Chapter 6).

2.3.1 Understanding and Improving Threshold-based Auto-labeling

One striking feature of TBAL is the promise that the auto-labeled data it outputs will have at least $1 - \epsilon_a$ accuracy, where ϵ_a is a user-given error tolerance parameter. The output labeled datasets are going to be used in multiple downstream applications, and thus, having such assurance on the quality of the dataset is highly valuable. Besides this promise, a high-level workflow and commercial success, little was known or understood about TBAL. Motivated by these factors, we undertook a systematic investigation to better understand and formalize the principles behind it. Our work made progress towards understanding and improving aspects of this technique. We discuss the main contributions below:

(i) Foundational understanding of TBAL. In Chapter 3 of this thesis, we provide the first in-depth theoretical and empirical analysis of TBAL. Based on our NeurIPS 2023 paper (Vishwakarma et al., 2023), this chapter establishes the foundations of TBAL, sheds light on the conditions under which TBAL can deliver on its promise, and its differences from techniques such as active learning and selective classification. One of the key findings of this work is that

TBAL may require a substantial amount of validation data to deliver on its promise of accuracy guarantees. TBAL uses validation data to find the threshold on the confidence scores. The points with scores above this threshold are auto-labeled. Naturally, the quantity of validation data plays a crucial role in finding the appropriate threshold and hence in the overall quality and quantity of the auto-labeled data – underestimation of the threshold can lead to higher auto-labeling errors, and overestimation can miss out on the points that could have been auto-labeled. We derive sample complexity bounds on the amount of human-labeled validation data required for guaranteeing accurate estimation of the auto-labeling errors for *any classifier* at *any threshold* on the given confidence function. The bounds suggest that in the *worst case without any data-specific assumptions* TBAL may require validation samples proportional to $\mathcal{O}(1/\epsilon_a^2)$ and the Rademacher complexity of the model class. This analysis sheds light on a hidden downside of TBAL systems, i.e., potentially prohibitive validation data usage. While this might be pessimistic in general, we discuss some potential solutions in section 2.5 that can bring down this cost in specialized settings.

(ii) Improving confidence functions for TBAL. The empirical analysis in Chapter 3 pointed to another key bottleneck in TBAL: the choice of confidence functions. This motivated a systematic study of these choices and potential improvements. Chapter 4 of this thesis highlights the limitations of common ad-hoc choices of confidence functions and introduces principled methods to address these shortcomings. We provide a brief summary of the contributions here. A common practice in TBAL is to train the model with Stochastic Gradient Descent (SGD) and use the softmax output of the model as confidence scores. The scores obtained from such a procedure are known to be overconfident—high scores for both correct and incorrect predictions (Nguyen et al., 2015a). Such scores can be detrimental to the performance of auto-labeling, and our experiments confirm this. Several methods have been introduced to overcome the overconfidence issue in neural networks, including calibration methods (Guo et al., 2017; Kumar et al., 2019; Moon et al., 2020; Gupta and Ramdas, 2022; Zhu et al., 2022; Hui et al., 2023). We evaluated both training-time and post-hoc calibration techniques in TBAL and found that it still misses out on significant coverage (proportion of auto-labeled points), since the calibration goal differs from

the auto-labeling objective. These results prompted the question: *What are the right choices of confidence functions for TBAL and how can we obtain such functions?* We answer this by proposing a framework to learn the right confidence functions for TBAL. In particular, we express the auto-labeling objective as an optimization problem over the space of confidence functions and the thresholds. In each round of TBAL, we solve this optimization as a post-hoc step after learning the classifier and use the solution (the learned confidence function) in the auto-labeling pipeline. Through extensive empirical evaluation on real data, we show that using the confidence scores obtained using our procedure boosts auto-labeling performance significantly in comparison to common choices of confidence functions and calibration methods.

2.3.2 Improving Pseudolabeling in Semi-Supervised Learning

A core challenge in pseudolabeling-based semi-supervised learning (SSL) is striking the right balance between the quality and quantity of pseudolabels. Many existing approaches (Lee et al., 2013; Sohn et al., 2020; Zhang et al., 2021b) rely on softmax confidence scores paired with hand-tuned thresholds to decide which predictions to treat as pseudolabels. However, recent studies (Loh et al., 2022; Mishra et al., 2024) have shown that such ad-hoc choices of scores often suffer from overconfidence and miscalibration, ultimately hurting the overall performance. Inspired by the ideas in TBAL, we introduce a more principled alternative: a framework that jointly learns both the confidence scores and thresholds in a way that explicitly accounts for a target error tolerance. This allows us to identify the largest set of pseudolabels that still satisfies a desired accuracy level, avoiding the pitfalls of prior approaches. When integrated into standard SSL pipelines, our method consistently improves model accuracy. We present the full method and results in Chapter 5.

2.3.3 Extending Weak Supervision to Structured Prediction.

Users often wish to use WS for *structured prediction*, where the output space consists of more than a binary or multi-class label set: e.g., rankings, graphs, manifolds, and more. In our work Shin et al. (2022b), we first lift the commonly used WS technique (triplet method) to *diverse*

label spaces such as rankings, hyperbolic spaces, etc. Next, in Vishwakarma and Sala (2022), we proposed a novel WS technique based on pseudo-euclidean embeddings (Goldfarb, 1985) and tensor decomposition (Anandkumar et al., 2014b), extending the scope of WS to labels in *any finite discrete metric spaces* and *constant curvature Riemannian manifolds*. We provide finite-sample error bounds for noise rate estimation in each scenario and generalization error guarantees for training downstream models on pseudolabels. Notably, the generalization error of these downstream models trained on potentially noisy labels scales as $\tilde{\mathcal{O}}(n^{-1/4})$, which is the same as the rates established for learning from clean labels in the structured prediction settings (Ciliberto et al., 2016). Here, n is the number of data points and the result holds as long as the weak supervision sources satisfy certain independence and symmetry assumptions. We evaluate our method empirically, confirming the theoretical results and showing improvements over Shin et al. (2022a). See Chapter 6 for further details.

2.4 Background

We provide brief background on these techniques and then summarize our results.

2.4.1 Threshold-based Auto-labeling

It is a promising solution for obtaining high-quality labeled data at low cost (SGT, 2022). It iteratively acquires human labels for a small portion of data, trains a model on the human-labeled data, then labels the remaining unlabeled data using this model's predictions above a *threshold* on the model's *confidence scores*. The threshold is determined using *validation data* to maximize the number of points labeled by the model (auto-labeled) while ensuring the fraction of erroneously auto-labeled points is below a user-given

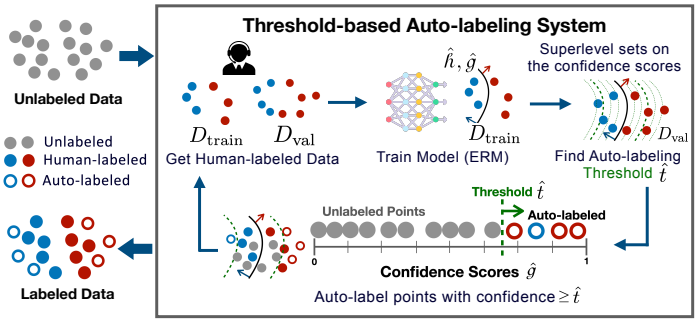


Figure 2.1: High-level diagram of TBAL system.

tolerance level ϵ_a.

Setting. In TBAL we are given a set of unlabeled points $X_u = \{\mathbf{x}_i\}_{i=1}^N$ drawn i.i.d from some distribution $P_{\mathcal{X}}$. We assume there is a latent groundtruth label y_i for each $\mathbf{x}_i$ and there is no ambiguity or randomness in this groundtruth. The algorithm can obtain the true labels by querying a groundtruth oracle (human) for chosen points. The goal is to auto-label (infer labels $\tilde{y}_i$) as many points as possible while minimizing the number of queries to the groundtruth oracle. We assume the label space $\mathcal{Y}$ is a discrete set of classes, i.e., $\mathcal{Y} = \{1, 2, \ldots, k\}$.

Auto-labeling Error and Coverage. These metrics measure the quality and quantity of the auto-labeled data output by TBAL. The auto-labeling coverage denotes the fraction of unlabeled points that were auto-labeled by TBAL. The auto-labeling error measures the fraction of auto-labeled points that were incorrectly labeled. More specifically, let $X_a \subseteq X_u$ be the set of auto-labeled points; then the coverage and error are defined as follows,

$$\widehat{\mathcal{E}} = \frac{1}{|X_a|} \sum_{\mathbf{x}_j \in X_a} \mathbb{1}(\tilde{y}_j \neq y_j) \quad \text{and} \quad \widehat{\mathcal{P}} = \frac{|X_a|}{|X_u|}. \tag{2.1}$$

Here $|\cdot|$ denotes set cardinality and $\mathbb{1}(\cdot)$ is the $0-1$ function.

Auto-labeling Error Constraint. Along with the unlabeled data, the user of TBAL also provides an error tolerance parameter $\epsilon_a \in (0, 1)$. TBAL is expected to output auto-labeled data such that $\widehat{\mathcal{E}} \leq \epsilon_a$.

Confidence functions. TBAL assumes there is a confidence function $g : \mathcal{X} \mapsto \Delta^k$ associated with classification model $h : \mathcal{X} \mapsto \mathcal{Y}$. Here Δ^k is a $k-1$-dimensional probability simplex. The function g gives access to model's confidence score $g(\mathbf{x})[\hat{y}]$ in its prediction $\hat{y} = h(\mathbf{x})$. As an example of g, consider the softmax outputs of the last layer of a neural network designed for a k-class classification problem.

TBAL workflow. The high-level steps are outlined as follows,

1. First, it creates two subsets D_{train} and D_{val} by drawing points randomly from X_u and querying their groundtruths. It also selects a model class $\mathcal{H}$.

2. On the D_{train} it trains a model $\hat{h}$ from the model class $\mathcal{H}$.

3. The model $\hat{h}$ is expected to be of low or moderate accuracy as it has been trained on a small amount of data. In this step, TBAL identifies the set of points on which the accuracy of $\hat{h}$ is at least $1-\epsilon_a$. For this, it finds a threshold $\hat{t}$ on the model's confidence scores $\hat{g}$ such that the auto-labeling error on the set of points with $\hat{g}(\mathbf{x})[\hat{y}] \geq \hat{t}$ is at most ϵ_a. Here, TBAL uses validation data D_{val} to estimate auto-labeling errors for each candidate threshold and selects the threshold that satisfies the error constraint.

4. Together, $(\hat{g}, \hat{t})$ identify the set of points $(\{\mathbf{x} \in X_u : \hat{g}(\mathbf{x}) \geq \hat{t}\})$ that are *safely* labeled using model's predictions $\hat{h}(\mathbf{x})$.

5. The auto-labeled points are removed from X_u, and similarly, the points in D_{train} falling in the auto-labeling region are removed as well.

6. If there are more unlabeled points left, TBAL queries groundtruth for a small batch of unlabeled points and adds them to the training set D_{train} and goes to step 2.

This procedure is adopted from a widely used commercial product, Amazon Sagemaker Groundtruth (SGT, 2022). One of the promising features of TBAL is the ability to provide strict control over the auto-labeling errors. Moreover, it is agnostic to the modalities of the inputs as long as we can train appropriate models. However, TBAL still requires a certain amount of labeled data from training and validation. Next, we describe a more general framework *weak supervision* that does not require any groundtruth labels.

2.4.2 Weak Supervision

Supervision (or labeling signal) could be available in various forms and at several levels of abstraction. For example, heuristic rules, pre-trained models, external knowledge bases, etc. These sources may not be highly accurate, but they contain useful signals about the groundtruth labels. Weak supervision (WS) (Ratner et al., 2016) is a rich set of techniques, which generate pseudolabels by *aggregating* such *cheaply available* but *potentially noisy* sources to produce accurate groundtruth labels. Several works have shown the success of WS for binary and

X (Emails)	Y (Latent)	λ_1	λ_2	λ_3	λ_4	λ_5
Dear customer, Click on this link to win lottery....	1	1	1	1	-1	-1
Dear abc, Subscribe to get instant relief from capitalism ...	1	-1	-1	1	1	1
Dear valued customer, Buy this GPU now to get 50% off..	1	-1	1	-1	1	1
Hey <name>, lets meet at the terrace in the evening.	-1	-1	-1	-1	-1	1

Figure 2.2: Illustration of outputs of labeling functions (observed) and groundtruth (latent) in weak supervision. Here 1 indicates positive and -1 negative labels, respectively.

multi-class classification (Ratner et al., 2016, 2018; Fu et al., 2020).

We refer to each such source as a *labeling function* (LF). These sources are just estimates of the ground truth answer for whatever task we are interested in. Figure 2.2 shows the outputs from the LFs. The instances (examples) are the emails. The column for the ground truth label Y is just a placeholder since we don't get to see it.

After observing the outputs of the sources, *the goal of weak supervision is to estimate the ground truth—and hopefully more accurately than any single source by itself!* A naive but reasonable first-cut way is to take the *majority vote* of the outputs for each point. This approach will work well when the LFs are independent and have similar qualities. However, some LFs could be more accurate and some noisier. This can make the majority vote less effective. Weak supervision approaches often model the distribution of the unobserved ground truth Y and source outputs $\lambda_1, \ldots, \lambda_m$ as an exponential noise model as follows,

$$P_\theta(\lambda_1, \lambda_2, \ldots \lambda_m, Y) = \frac{1}{Z} \exp\Big(\sum_{i=1}^{m} \theta_i \lambda_i Y\Big) \tag{2.2}$$

The θ parameters above encode how accurate each LF is, with a large θ_i indicating that the ith LF frequently agrees with Y, the ground truth. Learning these parameters is the major challenge in WS. A variety of methods have been proposed to this end (Dawid and Skene, 1979; Ratner et al., 2016; Bach et al., 2017; Varma et al., 2019; Sala et al., 2019; Fu et al., 2020). After learning these parameters, we can rely on the estimated model to infer the groundtruth labels. It turns out that we can use them to do *weighted aggregation, or, more concretely, given our model, to*

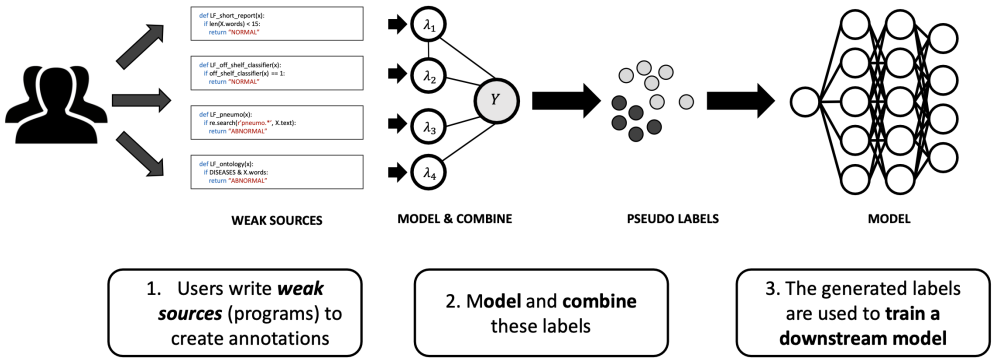


Figure 2.3: Weak supervision pipeline.

compute a posterior probability $P_{\hat{\theta}}(Y|\lambda_1, \ldots, \lambda_m)$. The overall pipeline of weak supervision is shown in Figure 6.1.

2.5 Outlook

This thesis makes progress toward alleviating the data bottleneck in machine learning, specifically by offering a principled understanding and advancement of threshold-based auto-labeling (TBAL) and by broadening the applicability of weak supervision (WS) to complex structured label spaces. These contributions not only clarify foundational aspects of widely used techniques but also raise several interesting challenges and open directions for future research.

Our analysis of TBAL revealed that its performance hinges on two major factors—the choice of confidence functions and the amount of validation data used. In this thesis, we settle the question of the right choice of confidence functions for TBAL by providing a principled solution (Colander). However, the challenge to reduce the validation data requirements remains open. While this may not be possible in general, under certain distributional assumptions and with techniques such as active testing (Farquhar et al., 2020), it may be feasible to conduct a tighter analysis and reduce the validation data requirements.

A basic assumption in TBAL, WS (and generally in semi-supervised learning) is that the unlabeled data is "in-distribution", i.e., all the data points belong to the pre-defined set of classes. However,

in practice, the unlabeled data may contain "out-of-distribution" (OOD) samples that do not belong to any of the classes (Cao et al., 2021). The presence of such inputs can seriously undermine the overall effectiveness of the data labeling algorithms we considered. In the other part of this thesis, we study techniques to ensure a given *well-trained* model is robust to OOD inputs at test time (Yang et al., 2022a). While it would be tempting to apply these procedures to tackle the issue of OOD points in the context of data labeling, the

A core assumption in TBAL, WS, and semi-supervised learning more broadly is that the unlabeled data comes from the same distribution as the labeled data—i.e., all samples belong to a predefined set of in-distribution classes. In practice, however, unlabeled data may contain out-of-distribution (OOD) inputs that do not belong to any known class (Cao et al., 2021). The presence of such inputs can significantly compromise the effectiveness of automated labeling methods. In the other part of this thesis, we study techniques to ensure a given *well-trained* model is robust to OOD inputs at test time (Yang et al., 2022a). The problem of detecting OOD inputs during the auto-labeling process is much more challenging, since we do not have a well-trained model to start with. This calls for new methods that can detect and abstain from labeling OOD inputs in the auto-labeling workflows.

Finally, as large language models (LLMs) continue to improve, their ability to assist in data labeling workflows presents new opportunities and risks (Huang et al., 2023b; Wang et al., 2024; Zhang et al., 2025a). While LLMs can drastically reduce the manual effort required for labeling, their susceptibility to hallucinations (Ji et al., 2023; Kalai and Vempala, 2024) and inherent biases (Kumar et al., 2025; Horych et al., 2025) raise concerns about the reliability of the labels they produce. Addressing these challenges is a fruitful direction for future research, with the potential to significantly enhance the scalability and reliability of data-labeling systems.

Chapter 3

Threshold-based Auto-labeling

Creating large-scale high-quality labeled datasets is a major bottleneck in supervised machine learning workflows. Threshold-based auto-labeling (TBAL), where validation data obtained from humans is used to find a confidence threshold above which the data is machine-labeled, reduces reliance on manual annotation. TBAL is emerging as a widely used solution in practice. Given the long shelf-life and diverse usage of the resulting datasets, understanding when the data obtained by such auto-labeling systems can be relied on is crucial. This is the first work to analyze TBAL systems and derive sample complexity bounds on the amount of human-labeled validation data required for guaranteeing the quality of machine-labeled data. Our results provide two crucial insights. First, reasonable chunks of unlabeled data can be automatically and accurately labeled by seemingly bad models. Second, a hidden downside of TBAL systems is potentially prohibitive validation data usage. Together, these insights describe the promise and pitfalls of using such systems. We validate our theoretical guarantees with extensive experiments on synthetic and real datasets.

3.1 Introduction

Machine learning (ML) models with millions or even billions of parameters are used to obtain state-of-the-art performance in many applications, e.g., object identification (Redmon and Farhadi,

2017), machine translation (Vaswani et al., 2017), and fraud detection (Zeng and Tang, 2021). Such large-scale models require training on large-scale labeled datasets. As an outcome, the typical supervised ML workflow begins with the construction of a large-scale, high-quality dataset. Datasets with up to millions of labeled data points have played a pivotal role in the advancement of computer vision. However, collecting labeled data is an expensive and time consuming process. A common approach is to rely on the services of crowdsourcing platforms such as Amazon Mechanical Turk (AMT) to get groundtruth labels.

Even with crowdsourcing (Raykar et al., 2010; Vinayak and Hassibi, 2016), obtaining labels for the entire dataset is expensive. To reduce costs, data labeling systems that partially rely on using a model's predictions as labels have been developed. Such systems date back to teacher-less training (Fralick, 1967). Modern examples include Amazon Sagemaker Ground Truth (SGT, 2022) and others (Superb-AI, 2022; Samsung-SDS, 2022; Airbus, 2022; Venguswamy et al., 2021). These approaches can be broadly termed *auto-labeling*.

Auto-labeling systems aim to label unlabeled data using predictions from ML models that are often trained on small amounts of human labeled data, which can produce incorrect labels. The shelf life of datasets is longer than that of models, e.g., ImageNet continues to be a benchmark for many computer vision tasks (Deng et al., 2009) fifteen years after its initial development. As a result, to reliably train new models on auto-labeled datasets and deploy them, we need a thorough understanding of how reliable the datasets output by these auto-labeling systems are. Unfortunately, many widely used commercial auto-labeling systems (SGT, 2022; Samsung-SDS, 2022) are largely opaque with limited public information on their functionality. It is therefore unclear whether the quality of the datasets obtained can be trusted. To address this, we study the high level workflow of a popular *threshold-based* auto-labeling (TBAL) system (see Figure 3.1). We emphasize that our goal is to understand such systems and their performance—not to promote them as a superior alternative to other approaches. Our goal is:

Goal. Develop a fundamental understanding of TBAL systems. This is crucial since there is a lack of theoretical understanding of the reliability of these systems despite their wide adoption.

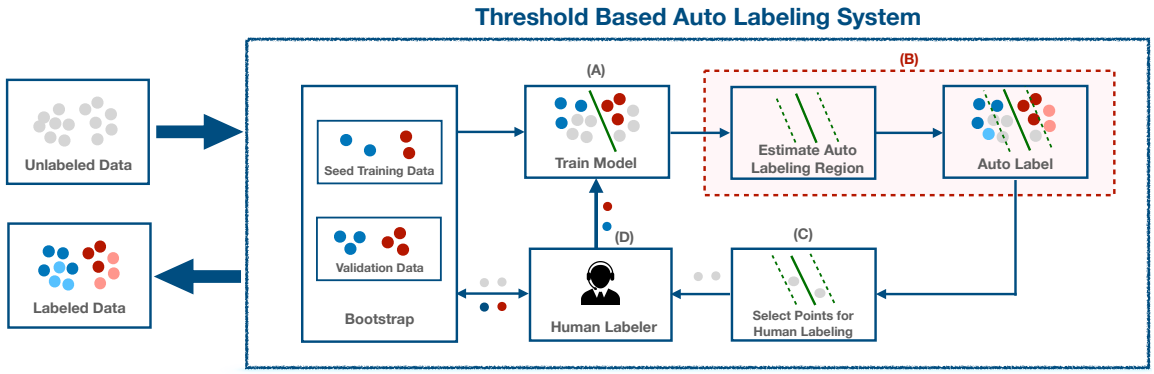


Figure 3.1: High-level workflow of threshold-based auto-labeling (TBAL). Box (B) shows the key component estimating the auto-labeling region using validation data and auto-labeling points in it.

The TBAL systems we study (Figure 3.1) work iteratively. At a high level, in each iteration, the system trains a model on currently available human labeled data and decides to label certain parts of unlabeled data using the trained model by finding high-accuracy regions using validation data. It then collects human labels on a small portion of unlabeled data that is deemed helpful for training the current model in the next iteration. The validation data is created by sampling i.i.d. points from the unlabeled pool and querying human labels for them. In addition to training data, the validation data is a major driver of the cost and accuracy of auto-labeling and will be a key component in our study.

Our Contributions. We study TBAL systems (Figure 3.1) and make the following contributions:

- Provide the **first theoretical characterization of TBAL systems**, developing tradeoffs between the quantity of manually labeled data and the quantity and quality of auto-labeled data (Section 3.3).
- Empirical results validating our theoretical understanding on real and synthetic data (Section 3.4).

Our results reveal **two important insights**. Promisingly, even poor quality models are capable of reliably labeling at least some data when we have access to sufficient validation data and a good confidence function that can accurately quantify the confidence of a given model on any

data point. On the downside, in certain scenarios, the quantity of the validation data required to reach a certain quantity and quality of auto-labeled data can be high.

3.2 Threshold-Based Auto-Labeling Algorithm

We begin with the problem setup and describe the TBAL algorithm that is inspired by the commercial systems (SGT, 2022). Then we provide experiments and theoretical analysis shedding light on the pros and cons of TBAL. We emphasize that TBAL is not our proposal, and our goal is to *understand* the effectiveness of such an auto-labeling system.

3.2.1 Problem Setup

Notation. Let the instance and label spaces be $\mathcal{X}$ and $\mathcal{Y} = \{1, \ldots, k\}$. We assume that there is some *deterministic* but unknown function $f^* : \mathcal{X} \mapsto \mathcal{Y}$ that assigns true label $y = f^*(\mathbf{x})$ to any $\mathbf{x} \in \mathcal{X}$. We also assume that there is a *noiseless oracle* $\mathcal{O}$ that can provide the true label $y \in \mathcal{Y}$ for any given $\mathbf{x} \in \mathcal{X}$. Let $X_{pool} \subseteq \mathcal{X}$ denote a sufficiently large pool of unlabeled data to be labeled.

The goal of an auto-labeling algorithm is to produce accurate labels $\tilde{y}_i \in \mathcal{Y}$ for points $\mathbf{x}_i \in X_{pool}$ while minimizing the number of queries to the oracle. Let $[m] := \{1, 2, \ldots, m\}$, $A \subseteq [N]$ be the set of indices of auto-labeled points, and $X_{pool}(A)$ be these points. The *auto-labeling error* $\widehat{\mathcal{E}}(X_{pool}(A))$ and the *coverage* $\widehat{\mathcal{P}}(X_{pool}(A))$ are defined as

$$\widehat{\mathcal{E}}(X_{pool}(A)) := \frac{1}{N_a} \sum_{i \in A} \mathbb{1}(\tilde{y}_i \neq f^*(\mathbf{x}_i)) \quad \text{and} \quad \widehat{\mathcal{P}}(X_{pool}(A)) := \frac{|A|}{N} = \frac{N_a}{N}, \tag{3.1}$$

where N_a denotes the size of the auto-labeled set A. TBAL algorithm aims to auto-label the dataset so that $\widehat{\mathcal{E}}(X_{pool}(A)) \leq \epsilon_a$ while maximizing coverage $\widehat{\mathcal{P}}(X_{pool}(A))$ for any given $\epsilon_a \in (0, 1)$.

Hypothesis Class and Confidence Function. A TBAL algorithm is given a fixed *hypothesis space* $\mathcal{H}$ and a *confidence function* $g : \mathcal{H} \times \mathcal{X} \mapsto T \subseteq \mathbb{R}^+$ that quantifies the confidence of $h \in \mathcal{H}$ on any data point $\mathbf{x} \in \mathcal{X}$. Confidence functions include prediction probabilities and margin scores.

For example, when $\mathcal{H}$ is a set of unit-norm homogeneous linear classifiers, i.e. $h_{\mathbf{w}}(\mathbf{x}) = \text{sign}(\mathbf{w}^T\mathbf{x})$ with $\mathbf{w} \in \{\mathbf{w} \in \mathbb{R}^d : ||\mathbf{w}||_2 = 1\}$, a reasonable confidence function is $g(h_{\mathbf{w}}, \mathbf{x}) = |\mathbf{w}^T\mathbf{x}|$.

Note that the target f^* might not be in the hypothesis space $\mathcal{H}$. Our analysis (Section 3.3) shows that the TBAL algorithm can work well, i.e., accurately label a reasonable fraction of unlabeled data with simpler hypothesis classes that do not contain the target hypothesis f^*. We illustrate this with a simple example in Section 3.2.3 and Figure 3.2. Note as well that the features $\mathbf{x}$ could be raw features or representations from self-supervised techniques, pre-trained models, etc. We analyze TBAL in settings (i) with no assumptions on the features and (ii) when the features are linearly separable.

3.2.2 Description of the Algorithm

The TBAL algorithm is given in Algorithm 1. It starts with an unlabeled pool X_{pool} and an auto-labeling error threshold ϵ. For ease of exposition, the algorithm is given the labeled validation set D_{val} of size N_v separately. In practice, it is created by selecting points at random from X_{pool}. The algorithm starts with an initial batch of n_s random data points and obtains oracle labels for these. The algorithm works in an iterative manner using the following steps.

1. Oracle labeled data obtained in each iteration i is added to the training pool $D^{(i)}_{train}$. It is used to train a model $\hat{h}_i$ by performing empirical risk minimization (ERM).

2. Find the region where $\hat{h}_i$ can auto-label accurately. The algorithm estimates a threshold $\hat{t}_i$ on the confidence score above which it can auto-label with the desired auto-labeling accuracy on the validation data (see Algorithm 2). Thresholds that have too little validation data are discarded, since their estimates are unreliable. The minimum threshold is found such that the sum of the estimated error $\widehat{\mathcal{E}}_a(\hat{h}_i, t|X^{(i)}_v)$ (see eq. (3.2)) and an upper confidence bound using the standard deviation of the estimated error, is at most the given auto-labeling error threshold.

3. Auto-label the points in the pool, $X^{(i)}_u$, which have confidence $g(\hat{h}_i, \mathbf{x}) \geq \hat{t}_i$. These are added to the set D_{out} and removed from the unlabeled pool. The validation points that fall in the

Algorithm 1: Threshold-based Auto-Labeling (TBAL)

Input: Unlabeled pool X_{pool}, auto labeling error threshold ϵ_a, seed data size n_s, batch size for active query n_b, labeled validation data pool D_{val}.

Output: $D_{out} = \{(\mathbf{x}_i, \tilde{y}_i) : \forall \mathbf{x}_i \in X_{pool}\}$

$X_u^{(1)} = X_{pool}; D_{val}^{(1)} = D_{val}$

$D_{query}^{(1)} = \texttt{randomly_query_batch}(X_u^{(1)}, n_s)$

Remove queried points from $X_u^{(1)}$

$D_{train}^{(0)} = \phi; i = 1; D_{out} = D_{out}^{(1)} = D_{query}^{(1)}$

while $X_u^{(i)} \neq \phi$ **do**

  $D_{train}^{(i)} = D_{train}^{(i-1)} \cup D_{query}^{(i)}$

  $\hat{h}_i = \texttt{empirical_risk_min}(\mathcal{H}, D_{train}^{(i)})$

  $\hat{t}_i$ = Estimate Threshold$(X_u^{(i)}, D_{val}^{(i)}, \epsilon_a, \hat{h}_i, n_0)$

  $D_{auto}^{(i)} = \{(\mathbf{x}, \hat{h}_i(\mathbf{x})) : \mathbf{x} \in X_u^{(i)}, g(\hat{h}_i, \mathbf{x}) \geq \hat{t}_i\}$

  $X_u^{(i+1)} = \{\mathbf{x} : \mathbf{x} \in X_u^{(i)}, g(\hat{h}_i, \mathbf{x}) < \hat{t}_i\}$

  $D_{val}^{(i+1)} = \{(\mathbf{x}, y) : (\mathbf{x}, y) \in D_{val}^{(i)}, g(\hat{h}_i, \mathbf{x}) < \hat{t}_i\}$

  $D_{query}^{(i+1)} = \texttt{active_query_batch}(\hat{h}_i, X_u^{(i+1)}, n_b)$

  Remove queried points from $X_u^{(i+1)}$

  $D_{out} = D_{out} \cup D_{auto}^{(i)} \cup D_{query}^{(i+1)}$

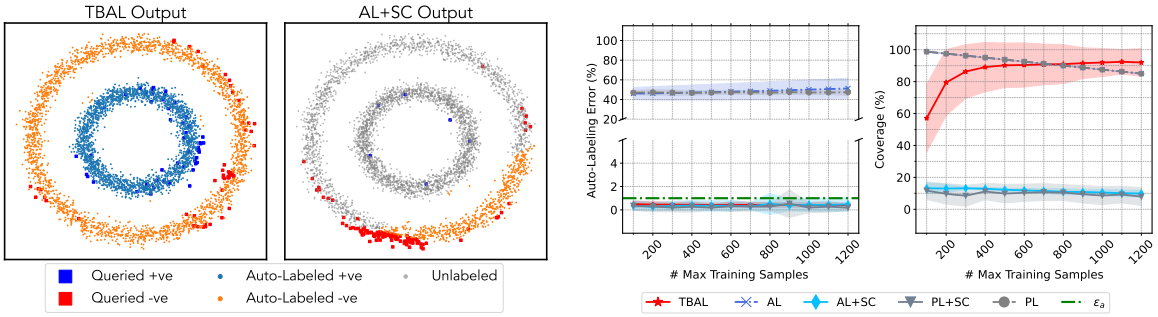
  $i = i + 1$

end while

auto-labeled region are also removed from the validation set so that in the next round, the validation set and the unlabeled pool are from the same region and the same distribution. Removing the auto-labeled points from X_{pool} is a crucial step in the TBAL algorithm that enables it to focus only on the remaining unlabeled regions in the next iteration.

4. If there are points left in X_{pool}, the algorithm selects points using some active querying strategy (Settles, 2009), obtains human labels for them, and adds them to the training pool. Note that the auto-labeled data is not added to the training set.

This process continues until there are no data points left to be labeled. The algorithm then outputs the labeled dataset, which is a mixture of human- and machine-labeled points.



(a) TBAL and AL+SC on Circles dataset (b) Auto labeling performance of various methods

Figure 3.2: Comparison of TBAL, active learning (AL) followed by selective classification (AL+SC) and passive learning (PL) followed by selective classification (PL+SC) on the Circles dataset (Sec. 3.2.3) using linear classifiers and confidence functions. (a) Samples auto-labeled, queried, and left unlabeled. (b) The auto-labeling error and coverage achieved by the algorithms. (50 trials.)

Algorithm 2: Estimate Threshold

Input: $X_u^{(i)}, D_{val}^{(i)}, \epsilon_a, \hat{h}_i, n_0$
Output: Threshold $\hat{t}_i$

1: $T_i = \{g(\hat{h}_i, \mathbf{x}) : (\mathbf{x}, y) \in D_{val}^{(i)}\}$
2: **for** $t \in T_i$ **do**
3: $\quad N_t = |\{\mathbf{x} \in X_v^{(i)} : g(\hat{h}_i, \mathbf{x}) \geq t\}|$
4: **end for**
5: $\tilde{T}_i = \{t : t \in T_i, N_t \geq n_0\} \cup \{\infty\}$
6: $\hat{t}_i = \min\{t \in \tilde{T}_i : \widehat{\mathcal{E}}_a(\hat{h}_i, t | X_v^{(i)}) + C_1 \hat{\sigma}_i \leq \epsilon_a\}$

3.2.3 Comparison between Auto-Labeling, Active Learning and Selective Classification

What is the difference between TBAL and methods such as active learning and selective classification?

Active learning. The goal of active learning (Settles, 2009) (AL) is to find the best model in hypothesis class $\mathcal{H}$ by training with less labeled data compared to passive learning. This is usually achieved by obtaining labels for the most informative points. Note that the *end goal is to output a model* from the function class whose predictions on new data are as good as the best model in the function class could.

Selective Classification. The goal of selective classification (SC) (El-Yaniv and Wiener, 2010) is to find the best combination of the hypothesis and selection functions to minimize error and maximize coverage of selection regions.

Auto-Labeling. The output of an auto-labeling procedure is a labeled dataset (not a model). When the hypothesis class is of lower complexity, it is often not possible to find a good classifier. The goal of an auto-labeling system is to label as much of the unlabeled data as accurately as possible with a given function class and with limited labeled data from humans.

Is active learning alone enough to auto-label data? AL has been found to be effective in reducing the number of labels needed to learn versus passive learning, particularly in low-noise cases (Hanneke, 2014). Doing auto-labeling using AL followed by SC may be effective in such settings. However, in real-world scenarios, noise levels may be higher and the hypothesis class could be misspecified. In these instances, using the model learned through active learning to automatically label all data may result in a high number of errors.

We illustrate this difference between AL, SC, and auto-labeling through an example. Suppose the data consists of two concentric circles, one for each class, with the same number of points per class (Figure 3.2(a)). This setting is not linearly separable. We run TBAL, AL, and AL followed by SC with an error tolerance of $\epsilon_a = 1\%$ and linear classifiers and confidence functions. The results are shown in Figure 3.2. Note that the multiple optimal linear classifiers will all incur an error of 50%. AL algorithms can only output models that make at least 50% error. If we naively use the output model for auto-labeling, we can obtain near full coverage but incur around 50% auto-labeling error. If we use the model output by AL with threshold-based SC, labeling error is reduced. However, it can only label $\approx 25\%$ of the unlabeled data. In contrast, TBAL can label almost all of the data accurately (close to 100% coverage) with less than 1% auto-labeling error.

3.3 Theoretical Analysis

The performance of the TBAL (Algorithm 1) depends on many factors, including the hypothesis class, the accuracy of the confidence function, the data sampling strategy, and the size of the

training and validation data. In particular, the amount of validation data plays a critical role in determining the accuracy of the confidence function, which in turn affects the accuracy and coverage.

We derive bounds on the auto-labeling error and the coverage for Algorithm 1 in terms of the size of the validation data, the number of auto-labeled points $N_a^{(k)}$, and the Rademacher complexity of the extended hypothesis class $\mathcal{H}^{T,g}$ induced by the confidence function g. Our first result, Theorem (3.3.1), applies to general settings and makes no assumptions on the particular form of the hypothesis class, the data distribution, and the confidence function. We then instantiate and specialize the results for a specific setting in Section 3.3.1. We introduce some notation to aid in stating our results,

Definition 3.3.1. *(Hypothesis Class with Abstain) The function g, set T and $\mathcal{H}$ induce an extended hypothesis class $\mathcal{H}^{T,g} := \mathcal{H} \times T$. Any function $(h,t) \in \mathcal{H}^{T,g}$ is defined as $(h,t)(\mathbf{x}) = h(\mathbf{x})$ if $g(h,\mathbf{x}) \geq t$ and $\perp$ otherwise. Here $(h,t)(\mathbf{x}) = \perp$ means (h,t) abstains in classifying the point $\mathbf{x}$.*

Error Definitions. Let $\mathcal{S} \subseteq \mathcal{X}$ denote a non-empty sub-region of $\mathcal{X}$ and $\mathrm{S}' \subseteq \mathcal{S}$ be a finite set of i.i.d. samples from $\mathcal{S}$. The subset $\mathcal{S}(h,t) \subseteq \mathcal{S}$ denotes the regions where (h,t) does not abstain i.e. $\mathcal{S}(h,t) := \{\mathbf{x} \in \mathcal{S} : (h,t)(\mathbf{x}) \neq \perp\}$, and the conditional probability mass associated with it is $\mathbb{P}(h,t \mid \mathcal{S}) := \mathbb{P}(\mathcal{S}(h,t) \mid \mathcal{S})$, and its empirical counterpart $\widehat{\mathbb{P}}(h,t \mid \mathrm{S}') := |\mathrm{S}'(h,t)|/|\mathrm{S}'|$. We use $\mathbb{P}(\mathcal{S})$ to denote the probability mass of set $\mathcal{S}$ and $\mathbb{P}(\mathcal{S}' \mid \mathcal{S})$ for the conditional probability of subset $\mathcal{S}' \subseteq \mathcal{S}$ given $\mathcal{S}$. The population level and empirical auto-labeling errors are defined as follows:

$$\begin{aligned} \mathcal{E}_a(h,t \mid \mathcal{S}) &:= \mathbb{E}_{\mathbf{x}|\mathcal{S}}[\ell_{0-1}^{\perp}(h,t,\mathbf{x},y)]/\mathbb{P}(h,t \mid \mathcal{S}), \\ \widehat{\mathcal{E}}_a(h,t \mid \mathrm{S}') &:= (\textstyle\sum_{\mathbf{x}_i \in \mathrm{S}'(h,t)} \ell_{0-1}^{\perp}(h,t,\mathbf{x}_i,y_i))/|\mathrm{S}'(h,t)| \end{aligned} \tag{3.2}$$

Here $\ell_{0-1}^{\perp}(h,t,\mathbf{x},y) := \ell_{0-1}(h,\mathbf{x},y) \cdot \ell_{\perp}(h,t,\mathbf{x})$ with $\ell_{0-1}(h,\mathbf{x},y) := \mathbb{1}(h(\mathbf{x}) \neq y)$, and $\ell_{\perp}(h,t,\mathbf{x}) := \mathbb{1}(g(h,\mathbf{x}) \geq t)$.

Rademacher Complexity. The Rademacher complexities for the function classes induced by the $\mathcal{H}, T, g$ and the loss functions are defined as $\mathfrak{R}_n\big(\mathcal{H}^{T,g}\big) := \mathfrak{R}_n\big(\mathcal{H}, \ell_{0-1}\big) + \mathfrak{R}_n\big(\mathcal{H}^{T,g}, \ell_{\perp}\big)$. Let

$\hat{h}_i$ and $\hat{t}_i$ be the ERM solution and the auto-labeling threshold at epoch i. Let $p_0 \in (0,1)$ be a constant such that $\mathbb{P}(\hat{h}_i, \hat{t}_i \mid \mathcal{X}^{(i)}) \geq p_0$ for all i. Let $X_v^{(i)}$ denote the validation set, and $n_v^{(i)}$ and $n_a^{(i)}$ the number of validation and auto-labeled points at epoch i. Let $\widehat{\mathcal{E}}_a(\hat{h}_i, \hat{t}_i \mid X_v^{(i)})$ be the empirical conditional risk of $\hat{h}_i$ in the region where $g(\hat{h}_i, \mathbf{x}) \geq \hat{t}_i$ evaluated on the validation data $X_v^{(i)}$.

We provide the following guarantees on the auto-labeling error and the coverage achieved by TBAL.

Theorem 3.3.1. *(Overall Auto-Labeling Error and Coverage) Let k denote the number of rounds of the TBAL Algorithm 1. Let $n_v^{(i)}, n_a^{(i)}$ denote the number of validation and auto-labeled points at epoch i and $n^{(i)} = |X^{(i)}|$. Let $X_{pool}(A_k)$ be the set of auto-labeled points at the end of round k. $N_a^{(k)} = \sum_{i=1}^k n_a^{(i)}$ be the total number of auto-labeled points. Then, for any $\delta \in (0,1)$, with probability at least $1-\delta$,*

$$\widehat{\mathcal{E}}\big(X_{pool}(A_k)\big) \leq \sum_{i=1}^k \frac{n_a^{(i)}}{N_a^{(k)}} \Bigg(\underbrace{\widehat{\mathcal{E}}_a\big(\hat{h}_i, \hat{t}_i \mid X_v^{(i)}\big)}_{(a)} + \underbrace{\frac{4}{p_0}\Big(\mathfrak{R}_{n_v^{(i)}}\big(\mathcal{H}^{T,g}\big) + \frac{2}{p_0}\sqrt{\frac{1}{n_v^{(i)}}\log\big(\tfrac{8k}{\delta}\big)}\Big)}_{(b)} \Bigg)$$
$$+ \underbrace{\frac{4}{p_0}\Bigg(\sum_{i=1}^k \frac{n_a^{(i)}}{N_a^{(k)}} \mathfrak{R}_{n_a^{(i)}}\big(\mathcal{H}^{T,g}\big) + \sqrt{\frac{k}{N_a^{(k)}}\log\big(\tfrac{8k}{\delta}\big)}\Bigg)}_{(c)}, \quad \textit{and}$$

$$\widehat{\mathcal{P}}\big(X_{pool}(A_k)\big) \geq \sum_{i=1}^k \mathbb{P}\big(\mathcal{X}^{(i)}(\hat{h}_i, \hat{t}_i)\big) - 2\mathfrak{R}_{n^{(i)}}\big(\mathcal{H}^{T,g}\big) - \sqrt{\tfrac{2k^2}{N}\log\big(\tfrac{8k}{\delta}\big)}.$$

Discussion. We interpret this result, starting with the auto-labeling error term $\widehat{\mathcal{E}}(X_{pool}(A_k))$. The term (a) $\widehat{\mathcal{E}}_a(\hat{h}_i, \hat{t}_i | X_v^{(i)})$ is the empirical conditional error in the auto-labeled region computed on the validation data in i-th round, which is at most ϵ_a. Thus, summing term (a) over all the rounds is at most ϵ_a. The term (b) provides an upper bound on the excess error over the empirical estimate term (a) as a function of the Rademacher complexity of $\mathcal{H}^{T,g}$ and the validation data used in each round. The last term (c) captures the variance in the overall estimate as a function of the total number of auto-labeled points and the Rademacher complexity of $\mathcal{H}^{T,g}$. If we let

$n_v^{(i)} \geq n_v$ i.e. the minimum validation points ensured in each round, then we can see the second term is $\mathcal{O}(\mathfrak{R}_{n_v}(\mathcal{H}^{T,g}))$ and the third term is $\mathcal{O}(\sqrt{1/n_v})$.

Therefore, validation data of size $\mathcal{O}\left(1/\epsilon_a^2\right)$ in each round is sufficient to get a $\mathcal{O}(\epsilon_a)$ bound on the excess auto-labeling error. The terms with Rademacher complexities suggest that it is better to use a hypothesis class and confidence function such that the induced hypothesis class has low Rademacher complexity. While such a hypothesis class might not be rich enough to include the target function, it would still be helpful for efficient and accurate auto-labeling of the dataset, which can then be used for training richer models in the downstream task. The coverage term provides a lower bound on the empirical coverage $\widehat{\mathscr{P}}(X_{pool}(A_k))$ in terms of the true coverage of the sequence of estimated hypotheses $\hat{h}_i$ and threshold $\hat{t}_i$.

We note that the size of the validation data needed to guarantee the auto-labeling error in each round by Algorithm 1 is optimal up to log factors. This follows by applying a result on the tail probability of the sum of independent random variables due to Feller (Feller, 1943):

Lemma 3.3.2. *Let c_1, c_2 and $\sigma > 0$. Let $\mathbf{x}_i \in X$ be a set of n i.i.d. points from $\mathcal{X}$ with corresponding true labels y_i. Given $(h,t) \in \mathcal{H}^{T,g}$, let $\mathbb{E}\left[\left(\ell_{0-1}^{\perp}(h,t,\mathbf{x}_i,y_i) - \mathcal{E}(h,t|\mathcal{X})\right)^2\right] = \sigma_i^2 > \sigma^2$ for every $\mathbf{x}_i$ for $\sigma_i > 0$ and let $\sum_i^n \sigma_i^2 \geq c_1$ then for every $\epsilon \in [0, (\sum_{i=1}^n \sigma_i^2)/\sqrt{c_1}]$ with $n_v < 12\sigma^2 \log(4c_2)/\epsilon^2$, the following holds w.p. at least $1/4$, $\mathcal{E}_a(h,t|\mathcal{X}) > \widehat{\mathcal{E}}_a(h,t|X) + \epsilon$.*

Therefore, if a sufficiently large validation set is not used in each round, there is a constant probability of erroneously deciding on a threshold for auto-labeling. Such a requirement on validation data also applies to active learning if we seek to validate the output model. Bypassing this requirement demands the use of approaches that are different from threshold-based auto-labeling and traditional validation techniques. We note the possibility of using recently proposed *active testing* techniques (Kossen et al., 2021), a nascent approach to reducing validation data usage.

3.3.1 Linear Classifier Setting

Next, we consider a simple setting where active learning is known to be optimal to see if TBAL can offer similar performance guarantees. To do so, we instantiate results from 3.3.1 to homogeneous linear separators under the uniform distribution in the realizable setting. Let $P_{\mathbf{x}}$ be supported on the unit ball in $\mathbb{R}^d$, $\mathcal{X} = \{\mathbf{x} \in \mathbb{R}^d : ||\mathbf{x}|| \leq 1\}$. Let $\mathcal{W} = \{\mathbf{w} \in \mathbb{R}^d : ||\mathbf{w}||_2 = 1\} = \mathbb{S}_d$, $\mathcal{H} = \{\mathbf{x} \mapsto \text{sign}(\langle \mathbf{w}, \mathbf{x} \rangle) \, \forall \mathbf{w} \in \mathcal{W}\}$, the score function be given by $g(h, \mathbf{x}) = g(\mathbf{w}, \mathbf{x}) = |\langle \mathbf{w}, \mathbf{x} \rangle|$, and set $T = [0, 1]$. For simplicity, we will use $\mathcal{W}$ in place of $\mathcal{H}$.

Corollary 3.3.2.1. *(Overall Auto-Labeling Error and Coverage) Let $\hat{\mathbf{w}}_i, \hat{t}_i$ be the ERM solution and the auto-labeling threshold, respectively, at epoch i. Let $n_v^{(i)}, n_a^{(i)}$ denote the number of validation and auto-labeled points at epoch i. Let the TBAL algorithm run for k-epochs. Then, for any $\delta \in (0, 1)$, w.p. at least $1 - \delta$,*

$$\widehat{\mathcal{E}}\big(X_{pool}(A_k)\big) \leq \sum_{i=1}^{k} \frac{n_a^{(i)}}{N_a^{(k)}} \Bigg(\underbrace{\widehat{\mathcal{E}}_a\big(\hat{\mathbf{w}}_i, \hat{t}_i \mid X_v^{(i)}\big)}_{(a)} + \underbrace{\frac{4}{p_0} \sqrt{\frac{2}{n_v^{(i)}} \left(2d \log\left(\frac{e n_v^{(i)}}{d}\right) + \log\left(\frac{8k}{\delta}\right)\right)}}_{(b)} \Bigg)$$

$$+ \underbrace{\frac{4}{p_0} \left(\sqrt{\frac{2k}{N_a^{(k)}} \left(2d \log\left(\frac{e N_a^{(k)}}{d}\right) + \log\left(\frac{8k}{\delta}\right)\right)} \right)}_{c}, \quad \textit{and}$$

$$\widehat{\mathcal{P}}\big(X_{pool}(A_k)\big) \geq 1 - \min_i \hat{t}_i \sqrt{4d/\pi} - 2k \sqrt{\frac{2}{N} \left(2d \log\left(\frac{eN}{d}\right) + \log\left(\frac{8k}{\delta}\right)\right)}.$$

These results imply that by ensuring the sum of the empirical validation error term (a) and the upper confidence interval to be less than ϵ_a in each round of the algorithm, we can ensure that the overall auto-labeling error remains below ϵ_a. Furthermore, by applying standard VC theory to the first round, we obtain that $\hat{t}_1 \leq 1/2$. Therefore, right after the first round, we are guaranteed to label at least half of the unlabeled pool. We empirically observe that TBAL has coverage at par with active learning while respecting the auto-labeling error constraint (See Figure 3.4(a)).

Tightness of the Bounds. We study this in the setting of the Unit-Ball experiment. The upper bound on excess risk in this setting is given in Corollary 3.3.2.1, which is an instantiation of

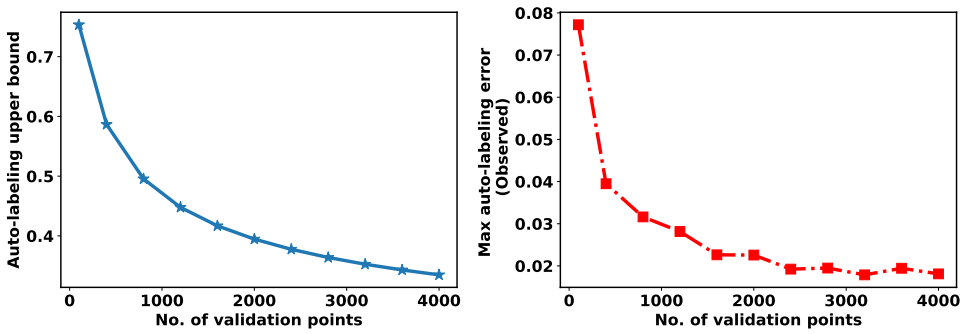


Figure 3.3: **Left:** Simplified upper bound from Corollary 3.4 (ignoring constants) on excess auto-labeling error for the Unit-Ball setting, i.e., homogeneous linear classifier with $d = 30$. **Right:** The worst observed auto-labeling error over 25 trials in the Unit-Ball experiment.

our general results to this specific setting. We consider a simplified form of the upper bound by ignoring the constants to get a sense of the rate in terms of the validation data size. We compute this simplified upper bound for different amounts of validation data. We compare these with the maximum auto-labeling error observed over 25 runs of auto-labeling in the Unit-Ball setting with different random seeds for each validation data size. The results are in Figure 3.3. As expected, we see that the worst-case error rate follows a similar rate as our upper bound, but the upper bound is conservative. Next, we explain why this is the case.

Our upper bound is slightly conservative, as it is based on a uniform bound over all hypotheses in a given hypothesis class. Since the individual hypotheses whose excess auto-labeling error we need to bound are not known a priori, we need to derive a bound on the number of validation samples using which we can guarantee that the excess auto-labeling error of any hypothesis (model) is small with high probability. Note that this is a conservative (worst-case) analysis to get an upper bound on the validation sample complexity. The upper bound has two parts: a) the Rademacher complexity of the hypothesis class, and b) a term with the number of validation samples. We note that on the validation samples, it matches lower bounds order-wise (see Lemma 3.3.2). This is the first analysis to provide these bounds based on uniform convergence without making any assumptions about the data distributions or hypothesis class. We provide further discussion on the role of Rademacher complexity in the Appendix A.2.1.

3.4 Experiments

We study the effectiveness of TBAL on synthetic and real datasets. We validate our theoretical results and aim to understand the amount of labeled validation and training data required to achieve a certain auto-labeling error and coverage. We also seek to understand whether our findings apply to real data—where labels may be noisy—along with how TBAL performs compared to common baselines.

Baselines. We compare TBAL to the following methods:

a) *Passive Learning (PL)* queries a subset of the points randomly to train a model from a given model class and then uses it to predict the labels for the remaining unlabeled pool.

b) *Active Learning (AL)* (using margin-random query strategy, described below) trains a model from a model class and uses it to predict the labels for the remaining unlabeled pool.

c) *Passive Labeling + Selective Classification (PL+SC)* first performs passive learning to train a model from a given model class. Then it performs auto-labeling on the unlabeled data using threshold-based selective classification with the model output by passive learning. Only those unlabeled points that are deemed as fit to be labeled by the selection function are auto-labeled.

d) *Active Learning + Selective Classification (AL+SC)* first performs active learning (using margin-random query strategy) to train a model from a given model class. It then performs auto-labeling using threshold-based selective classification with the model output by AL.

For selective classification in the above methods, we use Algorithm 2 to estimate the threshold and use it to perform auto-labeling. In experiments, we modify Algorithm 2 slightly—instead of estimating a single threshold for all classes, we estimate thresholds for each class separately.

Active Querying Strategy. We use the *margin-random* query strategy for querying the next batch of training data. In this strategy, the algorithm first sorts the points based on the margin (uncertainty) score and then selects the top Cn_b ($C > 1$) points from which n_b points are picked at random. This is a simple and computationally efficient method that balances the exploration and exploitation trade-off. We note that other active-querying strategies exist; we use margin-random

N_v	Error (%)		Coverage (%)	
	TBAL	AL+SC	TBAL	AL+SC
100	3.10 ±1.80	0.68 ±0.81	71.43 ±8.86	96.95 ±1.01
400	1.97 ±0.76	0.59 ±0.18	93.99 ±2.39	97.89 ±0.50
800	1.64 ±0.50	0.66 ±0.19	96.26 ±1.33	98.06 ±0.53
1200	1.39 ±0.39	0.67 ±0.19	96.67 ±0.84	98.10 ±0.45
1600	1.33 ±0.30	0.70 ±0.19	97.13 ±0.45	98.16 ±0.44
2000	1.28 ±0.34	0.71 ±0.21	97.15 ±0.54	98.20 ±0.34

N_v	Error (%)		Coverage (%)	
	TBAL	AL+SC	TBAL	AL+SC
100	3.10 ±1.80	0.68 ±0.81	71.43 ±8.86	96.95 ±1.01
400	1.65 ±0.65	0.32 ±0.15	93.27 ±2.50	96.91 ±0.99
800	1.08 ±0.47	0.24 ±0.16	96.01 ±1.16	96.31 ±1.36
1200	0.78 ±0.27	0.17 ±0.11	96.82 ±0.84	95.96 ±1.40
1600	0.65 ±0.20	0.13 ±0.08	96.93 ±0.57	95.70 ±1.38
2000	0.54 ±0.16	0.21 ±0.11	97.23 ±0.42	96.36 ±1.13

Table 3.1: **Unit-Ball.** Effect of variation of validation data size (N_v) with and without using a UCB on error estimates. We keep training data size N_q fixed at 500 and use error threshold $\epsilon_a = 1\%$. We report the mean and std. deviation over 10 runs with different random seeds. **Left**: with $C_1 = 0$. **Right**: with $C_1 = 0.25$.

N_v	Error (%)		Coverage (%)	
	TBAL	AL+SC	TBAL	AL+SC
200	4.77 ±0.18	3.35 ±0.80	83.14 ±3.65	78.53 ±7.05
400	4.57 ±0.26	3.53 ±0.73	90.70 ±3.11	86.39 ±5.11
600	4.32 ±0.17	3.70 ±0.63	92.96 ±0.46	88.90 ±4.83
800	4.66 ±0.20	3.84 ±0.70	92.42 ±0.89	88.67 ±3.88
1000	4.67 ±0.16	3.90 ±0.68	92.89 ±0.91	89.79 ±3.09

N_v	Error (%)		Coverage (%)	
	TBAL	AL+SC	TBAL	AL+SC
200	2.28 ±0.21	3.11 ±0.86	68.24 ±6.20	57.77 ±13.09
400	1.29 ±0.10	1.98 ±0.40	63.81 ±4.86	63.06 ±10.70
600	1.41 ±0.20	1.81 ±0.22	69.64 ±3.98	62.92 ±9.20
800	1.62 ±0.30	2.04 ±0.35	67.45 ±3.72	63.22 ±7.89
1000	1.64 ±0.23	1.97 ±0.26	70.28 ±2.82	66.11 ±8.00

Table 3.2: **IMDB.** Effect of variation of validation data size (N_v) with and without using a UCB on error estimates. We keep training data size N_q fixed at 500 and use error threshold $\epsilon_a = 5\%$. We report the mean and std. deviation over 10 runs with different random seeds. **Left:** with $C_1 = 0$. **Right:** with $C_1 = 0.25$.

as our standard querying strategy to keep the focus on comparing auto-labeling—not active learning approaches.

Datasets. We use the following synthetic and real datasets. We also provide empirical results on MNIST and another synthetic dataset in the Appendix. For each dataset, we split the data into two sufficiently large pools. One is used as X_{pool} on which auto-labeling algorithms are run and the other is used as X_{val} from which the algorithms subsample validation data.

a) *Unit-Ball* is a synthetic dataset of uniformly sampled points from the d-dimensional unit ball. The true labels are generated using a homogeneous linear separator with $\mathbf{w} = [1/\sqrt{d}, \ldots, 1/\sqrt{d}]$. We use $d = 30$ and generate $N = 20K$ samples, out of which $16K$ are in X_{pool} and $4K$ are in X_{val}. The dataset has just two classes but there is no margin between them.

N_v	Error (%)		Coverage (%)	
	TBAL	AL+SC	TBAL	AL+SC
2000	0.0 ±0.0	0.0 ±0.0	0.0 ±0.0	0.0 ±0.0
4000	13.88 ±5.42	13.31 ±10.79	0.72 ±0.55	0.48 ±0.04
6000	14.18 ±0.76	11.52 ±0.82	17.29 ±0.72	8.18 ±1.12
8000	13.97 ±0.14	11.31 ±0.51	36.36 ±1.78	23.40 ±1.15
10000	13.42 ±0.29	11.14 ±0.54	43.79 ±0.93	33.38 ±0.72

N_v	Error (%)		Coverage (%)	
	TBAL	AL+SC	TBAL	AL+SC
2000	0.0 ±0.0	0.0 ±0.0	0.0 ±0.0	0.0 ±0.0
4000	10.50 ±6.01	7.37 ±4.57	0.47 ±0.05	0.48 ±0.06
6000	10.61 ±0.62	7.71 ±1.03	10.16 ±1.10	4.31 ±1.10
8000	9.90 ±0.63	6.80 ±0.77	25.84 ±1.57	14.43 ±2.01
10000	8.97 ±0.36	6.87 ±0.48	32.19 ±1.34	21.96 ±1.35

Table 3.3: **Tiny-ImageNet.** Effect of variation of validation data size (N_v) with and without using a UCB on error estimates. We keep the training data size N_q fixed at $10k$ and use an error threshold $\epsilon_a = 10\%$. We report the mean and std. deviation over 5 runs with different random seeds. **Left:** with $C_1 = 0$. **Right:** with $C_1 = 0.25$.

b) *Tiny-ImageNet* (Le and Yang, 2015a) is a subset of the larger ImageNet (Deng et al., 2009) dataset, designed for image classification tasks. It consists of *200 classes*, each with 500 training images and 50 validation and test images. With a total of $100K$ images, Tiny ImageNet provides a diverse and challenging dataset. We use pre-computed embeddings of the images using CLIP (Radford et al., 2021).

c) *IMDB Reviews* (Maas et al., 2011) is a comprehensive collection of movie reviews, consisting of $50K$ individual reviews. It is a balanced dataset of positive and negative labels. We use the standard train set of size $25K$ and split it into X_{pool} and X_{val} of sizes $20K$ and $5K$ respectively. We compute embeddings of reviews using a pre-trained model `bge-large-en` (Xiao et al., 2023) from the Massive Text Embedding Benchmark (MTEB) (Muennighoff et al., 2022; Face, 2023).

d) *CIFAR-10* (Krizhevsky et al., 2009a) is an image dataset with 10 classes. We randomly split the standard training set into X_{pool} of size $40K$ and the validation pool of size $10K$. We use the raw features for training.

Models and Training. For the linear models, we use SVM with the usual hinge loss and train it to a loss tolerance 10^{-5}. To train a multi-layer perceptron (MLP) on the pre-computed embeddings of IMDB and Tiny-ImageNet, we use SGD with a learning rate of $0.05, 0.1$, respectively, and batch size of 64. To train the medium CNN we use SGD with a learning rate of 10^{-2}, batch size 256, and momentum of 0.9. More details on model training are in the Appendix.

N_q	Error (%)		Coverage (%)	
	TBAL	AL+SC	TBAL	AL+SC
200	1.67 ±0.29	2.15 ±0.45	73.30 ±3.49	57.17 ±11.09
400	1.63 ±0.19	1.61 ±0.29	72.59 ±3.16	64.53 ±16.61
600	1.67 ±0.21	1.83 ±0.30	71.38 ±2.15	70.50 ±5.68
800	1.67 ±0.27	1.90 ±0.31	69.10 ±4.51	65.74 ±10.14
1000	1.62 ±0.22	1.97 ±0.35	73.42 ±2.84	68.05 ±5.56

N_q	Error (%)		Coverage (%)	
	TBAL	AL+SC	TBAL	AL+SC
2000	9.22 ±1.04	7.42 ±0.71	17.51 ±1.16	9.33 ±0.66
4000	9.30 ±0.38	6.97 ±0.39	25.01 ±1.20	14.25 ±1.71
6000	9.12 ±0.22	6.85 ±0.26	28.06 ±0.75	17.51 ±0.36
8000	9.21 ±0.14	7.38 ±0.53	30.88 ±0.64	21.18 ±0.90
10000	8.95 ±0.23	7.10 ±0.26	32.31 ±1.21	22.34 ±0.61

Table 3.4: Results for varying N_q, the maximum number of samples the algorithm can use for training. **Left: IMDB** with $N_v = 1000, C_1 = 0.25, \epsilon_a = 5\%$. **Right: Tiny-ImageNet** with $N_v = 10K, C_1 = 0.25, \epsilon_a = 10\%$. Mean and std. deviations are reported.

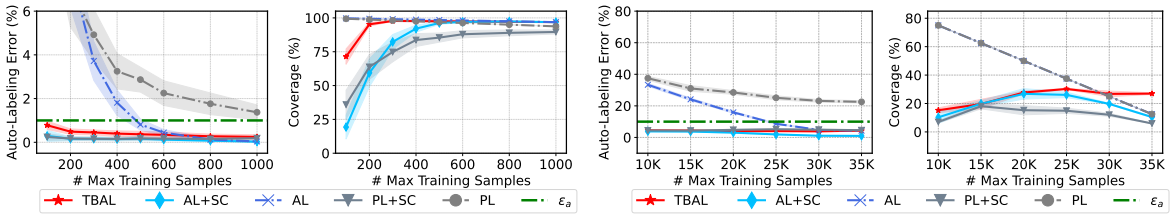
The score function g. For SVMs, we use the standard implementations of (Wu et al., 2003; Platt, 1999) in `sklearn` to get the prediction probabilities and use them as the score function. Neural networks use softmax output.

3.4.1 Role of Validation Data

The TBAL algorithm uses validation data to estimate the auto-labeling errors at various thresholds to determine the threshold for automatically labeling points accurately. Thus, it is crucial to have accurate estimates of the auto-labeling errors. Our analysis shows that to get such good estimates, large amounts of validation data are needed. In this section, we study the effect of varying the amount of validation data on auto-labeling performance.

Setup. We fix the maximum training data size N_q and run the algorithm with different amounts of validation data. We also consider the two cases where the algorithm uses an upper confidence bound on the error estimate and where it does not. We use the Unit-Ball, IMDB, Tiny-ImageNet, and datasets for this study with $N_q = 500, 500$, and $10K$ respectively, and the auto-labeling error thresholds $\epsilon_a = 1\%, 5\%, 10\%$, respectively. Initial seed data of size n_s is 20% of N_q, and query batch size n_b is 5% of N_q; $C = 2$ for both AL and TBAL for both datasets. We give the same initial seed samples of size n_s to all the methods to ensure they have the same starting point.

Results. Tables 3.1,3.2, and 3.3 demonstrate the impact of validation data on the performance of TBAL and other algorithms. The auto-labeling error and coverage of TBAL and other methods



(a) Unit-Ball: varying training samples size, validation samples size=$4K$, $\epsilon_a = 1\%, C_1 = 0.25$.

(b) CIFAR-10: varying training samples size,validation size=$10K$, $\epsilon_a = 10\%, C_1 = 0.25$.

Figure 3.4: Results for varying N_q, the maximum number of samples the algorithm can use for training while providing sufficient validation samples.

are affected by the amount of validation data provided. When the validation data is insufficient, the auto-labeling error of TBAL increases. However, as more validation data is used, the auto-labeling error and coverage of TBAL improve. Providing too little X_{val} can lead to incorrect estimates of the auto-labeling error, which in turn results in poor auto-labeling performance. This is further highlighted in our theoretical analysis as seen in Theorem 3.3.1. We also take a more nuanced look at the performance when the algorithm uses an upper confidence bound (with $C_1 = 0.25$) on the estimates and when it does not. We see the effects of not using any upper confidence bound (i.e., $C_1 = 0$). The left side tables in Tables 3.1,3.2, and 3.3 show the results when $C_1 = 0$, and the right side tables show the results when $C_1 = 0.25$. These show that not using UCB leads to high auto-labeling error (i.e., not meeting the guarantees) even when there is a sufficient amount of validation data. This can happen with high coverage as well—yielding a dataset with large errors. On the other hand, using UCB, i.e,. $C_1 = 0.25$, the algorithm can keep the auto-labeling error below the given threshold, but suffers in coverage.

In the above Tables, we omitted AL, PL, and PL+SC for clarity. Full results with all baselines are in Appendix A.3.3.

3.4.2 Role of Training Data Size

The labels queried for model training also play an important role in the process, while incurring costs to obtain. The next experiment focuses on the impact of training data on auto-labeling.

Setup. We limit the amount of training data the algorithm can use and record the resulting

auto-labeling error and coverage. We ensure all algorithms have sufficiently large but equal amounts of validation data. We run on Unit-Ball, IMDB, Tiny-Imagenet, and CIFAR-10 datasets with the same values of n_s, n_b, and C as in previous experiments.

Results. Figures 3.4(a), 3.4(b), and 3.2(b) indicate that TBAL and methods utilizing selective classification (AL+SC, PL+SC) maintain a high level of accuracy, even in scenarios where minimal training samples are used. This is expected as the threshold estimation method (when used with sufficient validation data) will find auto-labeling thresholds such that the auto-labeling error does not exceed ϵ_a. The impact of training data size can be seen clearly in the coverage achieved by the algorithms. As expected, with fewer training samples, the model has low accuracy, leading to low coverage. However, as more samples are acquired, a more accurate model within the function class is learned, resulting in increased coverage. The Appendix A.3 has additional discussion and results.

3.5 Related Works

In this section, we discuss works related to threshold-based auto-labeling and the underlying techniques, namely, active learning and selective classification.

There is a rich body of work on active learning on empirical and theoretical fronts (Settles, 2009; Dasgupta, 2011; Hsu, 2010; Hanneke, 2014; Citovsky et al., 2021; Ren et al., 2020). In active learning, the goal is to learn the best model in the given function class with fewer labeled data than in classical passive learning. To this end, various active learning algorithms have been developed and analyzed, e.g., uncertainty sampling (Tong and Koller, 2001; Mussmann and Liang, 2018), disagreement region based (Cohn et al., 1994; Hanneke, 2007), margin based (Balcan et al., 2007; Balcan and Long, 2013), importance sampling based (Beygelzimer et al., 2009) and others (Chaudhuri et al., 2015). Active learning has been shown to achieve exponentially smaller label complexity than passive learning in noiseless and low-noise settings (Dasgupta et al., 2005; Balcan et al., 2007; Hanneke, 2007, 2014; Balcan and Long, 2013; Dasgupta, 2006; Hsu, 2010; Chaudhuri et al., 2015; Krishnamurthy et al., 2017; Katz-Samuels et al., 2021). This suggests, in

these settings auto-labeling using active learning followed by selective classification is expected to work well. However, in practice, we do not have favorable noise conditions and the hypothesis class could be misspecified i.e. it may not contain the Bayes optimal classifier. In such cases, there are lower bounds on the label complexity of active learning that are order-wise the same as the label complexity for passive learning (Kääriäinen, 2006).

These findings have motivated more refined goals for active learning – abstain on hard to classify points and do well on the rest of the points. This idea is captured by the Chow's excess risk (Chow, 1970) and some of the recent works (Shekhar et al., 2021; Shah and Manwani, 2020; Puchkin and Zhivotovskiy, 2021; Zhu and Nowak, 2022) have proved exponential savings in label complexity for active learning when the goal is to minimize Chow's excess risk. The classifier learned by these methods is equipped with the abstain option and hence it can be readily applied for auto-labeling. However, the problem of misspecification of the hypothesis class still remains. Nevertheless, it would be interesting for future work to explore the connections between auto-labeling and active learning with abstention. Similar works on learning with abstention are done with passive learning as well (Cortes et al., 2016).

Selective classification (SC) is closely related to TBAL. In SC, the goal is to equip a given classifier with the option to abstain from the prediction to guarantee prediction quality. The foundations for selective classification are laid down in (El-Yaniv and Wiener, 2010; Wiener and El-Yaniv, 2011; El-Yaniv and Wiener, 2012; Wiener and El-Yaniv, 2015) where they give results on the error rate in the prediction region and the coverage of a given classifier. However, they lack practical algorithms to find the prediction region. A recent work (Gelbhart and El-Yaniv, 2019) proposes a new disagreement-based active learning strategy to learn a selective classifier.

Recent work studies a practical algorithm for threshold-based selective classification on deep neural networks (Geifman and El-Yaniv, 2017). The algorithm estimates the prediction threshold using training samples and they bound the error rate of the selective classifier using (Gascuel and Caraux, 1992). We note that their result is applicable to a specific setting of a given classifier. In contrast, in the TBAL algorithm analyzed in this paper, selective classification is done in each

round and the classifiers are not given a priori but instead learned via ERM on training data which is adaptively sampled in each round.

A closely related work (Qiu et al., 2023) studies an algorithm similar to TBAL for auto-labeling. Their emphasis is on the cost of training incurred when these systems use large-scale model classes for auto-labeling. They propose an algorithm to predict the training set size that minimizes the overall cost and provides an empirical evaluation.

Well-calibrated uncertainty scores are essential to the success of threshold-based auto-labeling. However, in practice, such scores are often hard to get. Moreover, neural networks can produce overconfident (unreliable) scores (Hein et al., 2018). Fortunately, there are plenty of methods in the literature to deal with this problem (Platt, 1999; Wu et al., 2003). More recently, various approaches have been proposed for uncertainty calibration for neural networks (Gawlikowski et al., 2021a; Minderer et al., 2021; Wang et al., 2021a; Krishnan and Tickoo, 2020; Ma et al., 2021; Seedat and Kanan, 2019). A detailed study of calibration methods and their impact on auto-labeling is beyond the scope of this work and left as future work.

Another line of work is emerging towards auto-labeling that does not rely on getting human labels but instead uses potentially noisy but cheaply available sources to infer labels (Ratner et al., 2016; Fu et al., 2020; Shin et al., 2022b; Vishwakarma and Sala, 2022). The focus of this paper, however, is on analyzing the performance of TBAL algorithms (SGT, 2022; Airbus, 2022) that have emerged recently as popular auto-labeling solutions.

3.6 Conclusions and Future Work

In this chapter, we analyzed threshold-based auto-labeling systems and derived sample complexity bounds on the amount of human-labeled validation data required to guarantee the quality of machine-labeled data. Our study shows that these methods can accurately label a reasonable size of data using seemingly bad models when good confidence functions are available. Our analysis points to the hidden downside of these systems in terms of a large amount of validation data usage and calls for more sample-efficient methods including active testing.

Chapter 4

Confidence Functions for Auto-labeling

Auto-labeling is an important family of techniques that produce labeled training sets with minimum manual annotation. A prominent variant, threshold-based auto-labeling (TBAL), works by finding thresholds on a model's confidence scores above which it can accurately automatically label unlabeled data. However, many models are known to produce overconfident scores, leading to poor TBAL performance. While a natural idea is to apply off-the-shelf calibration methods to alleviate the overconfidence issue, we show that such methods fall short. Rather than experimenting with ad-hoc choices of confidence functions, we propose a framework for studying the *optimal* TBAL confidence function. We develop a tractable version of the framework to obtain `Colander` (Confidence functions for Efficient and Reliable Auto-labeling), a new post-hoc method specifically designed to maximize performance in TBAL systems. We perform an extensive empirical evaluation of `Colander` and compare it against methods designed for calibration. `Colander` achieves up to 60% improvement on coverage over the baselines while maintaining error level below 5% and using the same amount of labeled data.

4.1 Introduction

The demand for labeled data in machine learning (ML) is perpetual. Obtaining it is expensive and time-consuming, creating a bottleneck in ML workflows. Threshold-based auto-labeling (TBAL)

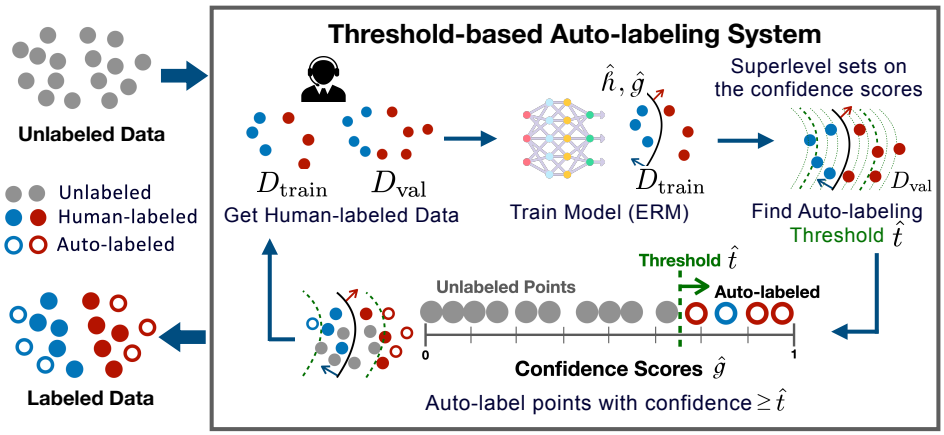


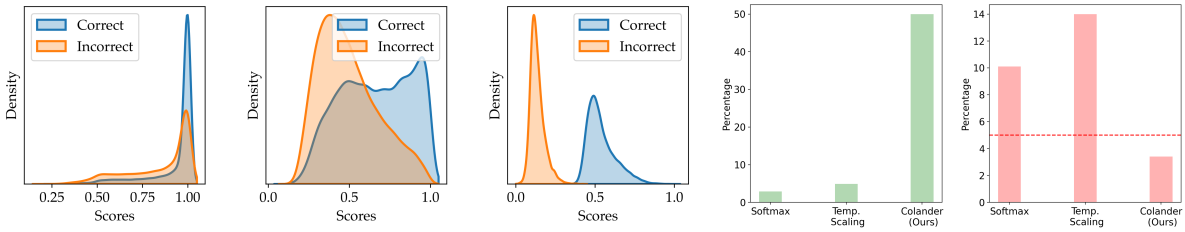
Figure 4.1: High-level diagram of TBAL system.

is a promising solution to obtain high-quality labeled data at low cost (SGT, 2022; Qiu et al., 2023; Vishwakarma et al., 2023). A TBAL system (Fig. 4.1) takes unlabeled data as input and outputs a labeled dataset. It works iteratively: in each iteration, it acquires human labels for a small chunk of data to train a model, then auto-labels points using the model's predictions where its *confidence scores* are above a certain threshold. The threshold is determined using validation data so that the auto-labeled points meet a desired *accuracy criteria*. The goal is to maximize *coverage*—the fraction of points automatically labeled (out of the total)—while maintaining accuracy. TBAL powers industry products like Amazon SageMaker Ground Truth SGT (2022).

The confidence function is critical to the TBAL workflow (Figure 4.1). Existing TBAL systems rely on common choices like softmax outputs from neural networks (Qiu et al., 2023; Vishwakarma et al., 2023). These functions *are not well aligned with the objective of the auto-labeling system.* Using them results in substantially suboptimal coverage (Figure 4.2(a)). For this reason, we ask:

What are the right choices of confidence functions for TBAL and how can we obtain them?

An ideal confidence function for auto-labeling will achieve the maximum coverage at a given auto-labeling error tolerance and thus will bring down the labeling cost significantly. Finding such an ideal function, however, is difficult because of the *inherent tension* between accuracy and



(a) Softmax (b) Temp. Scaling (c) `Colander` (Ours) (d) Coverage (e) Auto-labeling Err.

Figure 4.2: Scores distributions (Kernel Density Estimates) of a CNN model trained on CIFAR-10 data. (a) softmax scores of the vanilla training procedure (SGD) (b) scores after post-hoc calibration using temperature scaling, and (c) scores from our `Colander` procedure applied on the same model. For training the CNN model, we use 4000 points drawn randomly and 1000 validation points (of which 500 are used for Temp. Scaling and `Colander`). The test accuracy of the model is 55%. Figures (d) and (e) show the coverage and auto-labeling error of these methods. The dotted-red line corresponds to a user-given error tolerance of 5%.

coverage. The models used in auto-labeling are often highly inaccurate so achieving a certain error guarantee is easier when being conservative in terms of confidence—but this reduces coverage. Conversely, high coverage may appear to require lowering the requirements in confidence, but this may easily lead to overshooting the desired error level. This is compounded by the fact that TBAL is iterative, so even small deviations in error levels can cascade in future iterations.

Overconfidence may further stymie hopes of balancing accuracy and coverage. While overconfidence is a challenge in general, it is exacerbated in TBAL, since models are trained on a small amount of labeled data, they are often inaccurate, making the problem of designing confidence functions even more challenging. Common choices produce overconfident scores, i.e., high scores for both correct and incorrect predictions (Szegedy et al., 2014; Nguyen et al., 2015a; Hendrycks and Gimpel, 2017a; Hein et al., 2018; Bai et al., 2021). Fig. 4.2(a) shows that softmax scores are overconfident, resulting in poor auto-labeling performance.

Several methods have been introduced to address overconfidence, including a variety of calibration techniques (Guo et al., 2017). Applying these can nevertheless miss out on significant performance (Figure 4.2(b)) since the calibration goal differs from auto-labeling. From the auto-labeling standpoint, we seek minimum overlap between the correct and incorrect model prediction scores.

Other approaches (Corbière et al., 2019; Moon et al., 2020) bake the objective of separating scores into model training or use different optimization procedures (Zhu et al., 2022) that encourage separation. We observe that these do not help TBAL either, since, after some point, the model is correct on almost all the training points, making it hard to train it to discriminate between its own correct and incorrect predictions.

We tackle these challenges by ***proposing a framework to learn suitable confidence functions*** for TBAL. In particular, we express the auto-labeling objective as an optimization problem over the space of confidence functions and thresholds. Our framework subsumes existing methods, i.e., they are points in the space of solutions. The resulting method, `Colander` (Confidence functions for Efficient and Reliable Auto-labeling), relies on a practical surrogate to the framework that can be used to learn optimal confidence functions for auto-labeling. Using these learned functions in TBAL can achieve up to 60% improvement in coverage versus baselines like softmax, temperature scaling (Guo et al., 2017), CRL (Moon et al., 2020), and FMFP (Zhu et al., 2022). We summarize our contributions as follows,

1. We propose a principled framework to study the choices of confidence functions suitable for auto-labeling and provide a practical method (`Colander`) to learn confidence functions for efficient and reliable auto-labeling.
2. We systematically study commonly used choices of scoring functions and calibration methods and demonstrate that they lead to poor auto-labeling performance.
3. Through extensive empirical evaluation on real-world datasets, we show that using the confidence scores obtained using our procedure boosts auto-labeling performance significantly in comparison to common choices of confidence functions and calibration methods.

4.2 Background and Motivation

We provide notation, background on TBAL and its relationship to other methods, and describe the importance of confidence functions.

Notation. Let $[m] := \{1, 2, \ldots, m\}$ for any natural number m. Let X_u be a set of unlabeled points drawn from some instance space $\mathcal{X}$. Let $\mathcal{Y} = \{1, \ldots, k\}$ be the label space. There is an unknown ground truth labeling function $f^* : \mathcal{X} \to \mathcal{Y}$. Let $\mathcal{O}$ be a *noiseless* oracle that provides the true label for any point $\mathbf{x} \in \mathcal{X}$. Denote the model (hypothesis) class by $\mathcal{H}$, where each $h \in \mathcal{H}$ is a function $h : \mathcal{X} \to \mathcal{Y}$. Each classifier h also has an associated *confidence function* $g : \mathcal{X} \to \Delta^k$ that quantifies the confidence of the prediction by model $h \in \mathcal{H}$ on any data point $\mathbf{x} \in \mathcal{X}$. Here, Δ^k is a $(k-1)$-dimensional probability simplex. Let $\mathbf{v}[i]$ denote the i^{th} component for any vector $\mathbf{v} \in \mathbb{R}^d$. For any point $\mathbf{x} \in \mathcal{X}$ the prediction is $\hat{y} := h(\mathbf{x})$ and the associated confidence is $g(\mathbf{x})[\hat{y}]$. The vector $\mathbf{t}$ denotes scores over k-classes, and $\mathbf{t}[y]$ denotes its y^{th} entry, i.e., score for class y. Table B.1 (in Appendix B.2.4) contains a summary of the notation.

Threshold-based auto-labeling. It seeks to obtain labeled datasets while reducing the labeling burden on humans (Figure 4.1). The input is a pool of unlabeled data X_u. It outputs, for each $\mathbf{x} \in X_u$, a label $\tilde{y} \in \mathcal{Y}$. The output label could be either y, from the oracle (human), or $\hat{y}$, from the model. Let N_u be the number of unlabeled points, $A \subseteq [N_u]$ the set of indices of auto-labeled points, and $X_u(A)$ be these points. Let N_a be the size of the auto-labeled set A. The *auto-labeling error*, denoted by $\widehat{\mathcal{E}}(X_u(A))$, and the *coverage*, denoted by $\widehat{\mathcal{P}}(X_u(A))$, are defined as follows:

$$\widehat{\mathcal{E}}(X_u(A)) := \frac{1}{N_a} \sum_{i \in A} \mathbb{1}(\tilde{y}_i \neq f^*(\mathbf{x}_i)), \quad \text{and} \quad \widehat{\mathcal{P}}(X_u(A)) := |A|/N_u = N_a/N_u. \tag{4.1}$$

The goal of an auto-labeling algorithm is to label the dataset so that $\widehat{\mathcal{E}}(X_u(A)) \leq \epsilon_a$ while maximizing coverage $\widehat{\mathcal{P}}(X_u(A))$ for a *user-given error tolerance* parameter $\epsilon_a \in [0, 1]$. As depicted in Figure 4.1, the TBAL algorithm proceeds iteratively. In each iteration, it queries labels for a subset of unlabeled points from the oracle. It trains a classifier from the model class $\mathcal{H}$ on the oracle-labeled data acquired till that iteration. It then uses the model's confidence scores on the validation data to identify the region in the instance space, where the current classifier is confidently accurate and automatically labels the points in this region. The auto-labeled points are removed from the unlabeled pool. Similarly, to maintain parity between the validation and

unlabeled data in the next round, the validation points in the auto-labeling region are removed as well. These steps are executed in a loop until all the data is labeled or the budget to query oracle labels is exhausted.

Fundamental differences between TBAL, self-training, and active learning. At first glance, TBAL may appear similar to active learning (AL) Settles (2009), self-training (ST) Amini et al. (2023), and selective classification (SC) Geifman and El-Yaniv (2017). However, as described in Vishwakarma et al. (2023), it is a fundamentally different technique designed with different goals. Perhaps the most substantial difference is that TBAL's aim is to create accurately labeled datasets, while the goal in AL and ST is to learn the best (in terms of generalization error) possible classifier in a given model class with limited ground truth labels. This difference is most substantial in the settings where AL converges to a bad classifier, e.g., due to incorrect choice of the model class, sampling bias, etc. Vishwakarma et al. (2023) illustrates this notion with a scenario where TBAL coverage is above 95% while the other techniques average around 20%. See Appendix B.1.1 for details.

Problems with confidence functions in TBAL. The success of TBAL hinges on the ability of the classifier's confidence scores to distinguish between correct and incorrect labels. Prior works on TBAL (Vishwakarma et al., 2023; Qiu et al., 2023) train the model with stochastic gradient descent (SGD) and use the softmax output of the model as confidence scores, which are known to be overconfident (Nguyen et al., 2015a). A natural choice to mitigate this problem is to use post-hoc calibration techniques, e.g., temperature scaling (Guo et al., 2017). We evaluate these choices by running TBAL for a single round on the CIFAR-10 (Krizhevsky et al., 2009a) dataset with a SimpleCNN model with 5.8M parameters (Hussain, 2021) with error threshold 5%. Details are in Appendix B.1.2.

In Figures 4.2(d) and 4.2(e) we observe that using softmax scores from the classifier only produces 2.9% coverage while the error threshold is violated with 10% error. Using temperature scaling only increases the coverage marginally to 4.9% and still violates the threshold with error 14%. Looking closer at the scores for correct versus incorrect examples on validation data, we observe

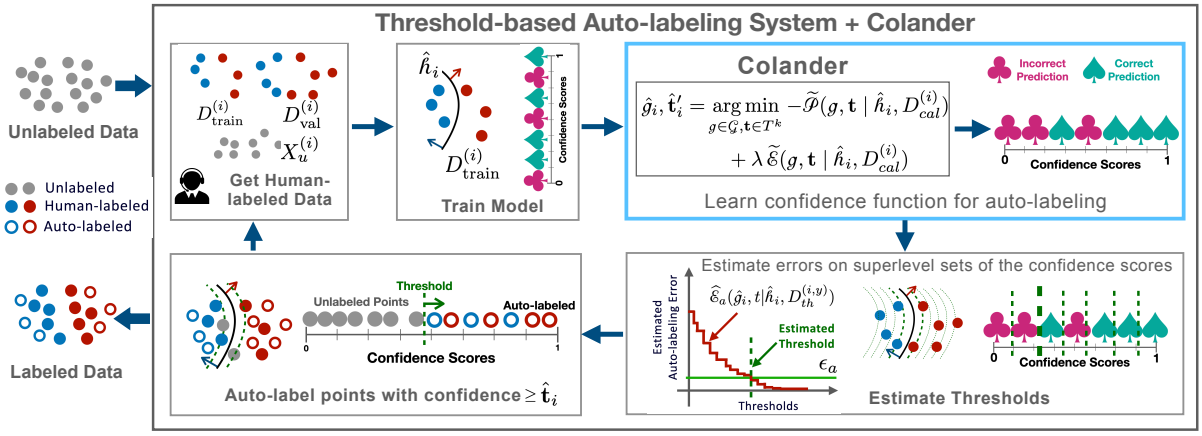


Figure 4.3: Threshold-based Auto-labeling with `Colander`: takes unlabeled data as input, selects a small subset of data points, and obtains human labels for them to create $D^{(i)}_{\text{train}}$ and $D^{(i)}_{\text{val}}$ for the ith iteration. Trains model $\hat{h}_i$ on $D^{(i)}_{\text{train}}$. In contrast to the standard TBAL procedure, here we randomly split $D^{(i)}_{\text{val}}$ into two parts, $D^{(i)}_{\text{cal}}$ and $D^{(i)}_{\text{th}}$. `Colander` kicks in, takes $\hat{h}_i$ and $D^{(i)}_{\text{cal}}$ as input and learns a coverage maximizing confidence function $\hat{g}_i$ for $\hat{h}_i$. Using $D^{(i)}_{\text{th}}$ and $\hat{g}_i$ auto-labeling thresholds $\hat{\mathbf{t}}_i$ are determined to ensure the auto-labeled data has error at most ϵ_a (a user-given parameter). After obtaining the thresholds, the rest of the steps are the same as standard TBAL. The whole workflow runs until all the data is labeled or another stopping criterion is met.

a large overlap for softmax (Figure 4.2(a)) and a marginal shift with considerable overlap for temperature scaling (Figure 4.2(b)). To overcome this challenge, we propose a novel framework (Section 4.3) to learn such confidence functions in a principled way. Our method in this example can achieve 50% coverage with an error of 3.4% within the desired threshold (Figure 4.2(c)).

4.3 Proposed Method (`Colander`)

The observations in Figure 4.2(a) and 4.2(b) suggest that fixed choices of confidence functions can leave significant coverage on the table. To find a better choice in a principled manner, we develop a framework based on auto-labeling objectives—maximizing coverage while having bounded auto-labeling error. We instantiate it by using empirical estimates and easy-to-optimize surrogates. We use the overall TBAL workflow from Vishwakarma et al. (2023) and introduce our method to replace the confidence (scoring) function after training the classifier.

4.3.1 Auto-labeling optimization framework

In any iteration of TBAL, we have a model $\hbar$ trained on a subset of data labeled by the oracle. This model may not be highly accurate. However, it could be accurate in some regions of the instance space, and with the help of a confidence function g, we want to identify the points where the model is correct and auto-label them. As we saw earlier, arbitrary choices of g perform poorly on this task. Instead, we propose a framework to find the right function from a sufficiently rich family.

Optimal confidence function. To find a confidence function aligned with our objective, we consider a space of thresholds T (e.g. $[0, 1]$) and functions $\mathcal{G} : \mathcal{X} \to T^k$, where T^k is the $k-$dimensional product space of T. We express the auto-labeling objective as an optimization problem (P4.1):

$$\underset{g \in \mathcal{G}, \mathbf{t} \in T^k}{\arg\max} \quad \mathscr{P}(g, \mathbf{t} \mid \hbar) \;\; \text{s.t.} \;\; \mathscr{E}(g, \mathbf{t} \mid \hbar) \leq \epsilon_a. \tag{P4.1}$$

Here $\mathscr{P}(g, \mathbf{t}|\hbar)$ and $\mathscr{E}(g, \mathbf{t} \mid \hbar)$ are the population-level coverage and auto-labeling error, which are defined as follows,

$$\mathscr{P}(g, \mathbf{t} \mid \hbar) := \mathbb{P}_{\mathbf{x}}\big(g(\mathbf{x})[\hat{y}] \geq \mathbf{t}[\hat{y}]\big) \quad \text{and} \quad \mathscr{E}(g, \mathbf{t} \mid \hbar) := \mathbb{P}_{\mathbf{x}}\big(y \neq \hat{y} \mid g(\mathbf{x})[\hat{y}] \geq \mathbf{t}[\hat{y}]\big). \tag{4.2}$$

The optimal $g^\star$ and $\mathbf{t}^\star$ that achieve the maximum coverage while satisfying the auto-labeling error constraint belong to the solution(s) of this optimization problem.

4.3.2 Practical method to learn confidence functions

The framework provides a theoretical characterization of the optimal confidence functions and thresholds for TBAL. However, it is impractical since the distributions and $f^\star$ are unknown. Next, we give a practical method based on the above framework to learn confidence functions for TBAL.

Empirical optimization problem. Since we do not know the distributions of $\mathbf{x}$ and $f^\star$, we use estimates of coverage and auto-labeling errors on a fraction of validation data to solve the

optimization problem. Let D be some finite number of labeled samples, and then the empirical coverage and auto-labeling error are defined as follows,

$$\widehat{\mathscr{P}}(g, \mathbf{t} \mid \hbar, D) := \frac{1}{|D|} \sum_{(\mathbf{x},y)\in D} \mathbb{1}\big(g(\mathbf{x})[\hat{y}] \geq \mathbf{t}[\hat{y}]\big), \tag{4.3}$$

$$\widehat{\mathscr{E}}(g, \mathbf{t} \mid \hbar, D) := \frac{\sum_{(\mathbf{x},y)\in D} \mathbb{1}\big(y \neq \hat{y} \wedge g(\mathbf{x})[\hat{y}] \geq \mathbf{t}[\hat{y}]\big)}{\sum_{(\mathbf{x},y)\in D} \mathbb{1}\big(g(\mathbf{x})[\hat{y}] \geq \mathbf{t}[\hat{y}]\big)}. \tag{4.4}$$

We randomly split the validation data into two parts D_{cal} and D_{th} and use D_{cal} to compute $\widehat{\mathscr{P}}(g, \mathbf{t} \mid \hbar, D_{\text{cal}})$ and $\widehat{\mathscr{E}}(g, \mathbf{t} \mid \hbar, D_{\text{cal}})$. Using these estimates, we now seek to solve the following problem,

$$\max_{g\in\mathcal{G}, \mathbf{t}\in T^k} \quad \widehat{\mathscr{P}}(g, \mathbf{t} \mid \hbar, D_{\text{cal}}) \;\; \text{s.t.} \;\; \widehat{\mathscr{E}}(g, \mathbf{t} \mid \hbar, D_{\text{cal}}) \leq \epsilon_a. \tag{P4.2}$$

Nevertheless, the presence of 0-1 variables means the problem remains challenging.

Surrogate optimization problem. To make the optimization (P4.2) tractable using gradient-based methods, we introduce differentiable surrogates for the 0-1 variables. Let $\sigma(\alpha, z) := 1/(1 + \exp(-\alpha z))$ denote the sigmoid function on $\mathbb{R}$ with scale parameter $\alpha \in \mathbb{R}$. It is easy to see that, for any g, y and $\mathbf{t}$, $g(\mathbf{x})[y] \geq \mathbf{t}[y] \iff \sigma(\alpha, g(\mathbf{x})[y] - \mathbf{t}[y]) \geq 1/2$. Using this fact, we define the following surrogates of the auto-labeling error and coverage:

$$\widetilde{\mathscr{P}}(g, \mathbf{t} | \hbar, D_{\text{cal}}) := \frac{1}{|D_{\text{cal}}|} \sum_{(\mathbf{x},y)\in D_{\text{cal}}} \sigma\big(\alpha, g(\mathbf{x})[\hat{y}] - \mathbf{t}[\hat{y}]\big), \tag{4.5}$$

$$\widetilde{\mathscr{E}}(g, \mathbf{t} \mid \hbar, D_{\text{cal}}) := \frac{\sum_{(\mathbf{x},y)\in D_{\text{cal}}} \mathbb{1}\big(y \neq \hat{y}\big)\, \sigma\big(\alpha, g(\mathbf{x})[\hat{y}] - \mathbf{t}[\hat{y}]\big)}{\sum_{(\mathbf{x},y)\in D_{\text{cal}}} \sigma\big(\alpha, g(\mathbf{x})[\hat{y}] - \mathbf{t}[\hat{y}]\big)}, \tag{4.6}$$

and the surrogate optimization problem as follows,

$$\underset{g \in \mathcal{G}, \mathbf{t} \in T^k}{\arg\min} \quad -\widetilde{\mathcal{P}}(g, \mathbf{t} \mid \hbar, D_{\text{cal}}) + \lambda\, \widetilde{\mathcal{E}}(g, \mathbf{t} \mid \hbar, D_{\text{cal}}) \tag{P4.3}$$

Here, $\lambda \in \mathbb{R}^+$ is the penalty term controlling the relative importance of the auto-labeling error and coverage. We tune it with the procedure discussed in Section 4.4.3. The gap between the surrogate and actual coverage diminishes as $\alpha \to \infty$. We discuss this in the Appendix.

Choice of $\mathcal{G}$. Our framework is flexible with respect to the choice of function class $\mathcal{G}$. In this work, we use neural networks with at least two layers on the model class $\mathscr{H}$. We use representations from the last two layers as input for the functions in $\mathcal{G}$ (Figure 4.4). Let $\mathbf{z}^{(1)}(\mathbf{x}; \hbar) \in \mathbb{R}^k$ and $\mathbf{z}^{(2)}(\mathbf{x}; h) \in \mathbb{R}^{d_2}$ be the outputs of the last and the second-last layer of the net h for input $\mathbf{x}$ and let $\mathbf{z}(\mathbf{x}; h) := [\mathbf{z}^{(1)}(\mathbf{x}; \hbar), \mathbf{z}^{(2)}(\mathbf{x}; \hbar)]$ denote the concatenation. This input is passed to network $\mathcal{G}_{nn_2} : \mathbb{R}^{k+d_2} \mapsto \Delta^k$; it outputs confidence scores for the k classes. Specifically g is defined as $g(\mathbf{x}) := \texttt{softmax}\big(\mathbf{W}_2 \texttt{tanh}(\mathbf{W}_1 \mathbf{z}(\mathbf{x}; \hbar))\big)$. Here $\mathbf{W}_1 \in \mathbb{R}^{(k+d_2) \times 2(k+d_2)}$ and $\mathbb{R}^{2(k+d_2) \times k}$ are the learnable weight matrices. As usual, for $\mathbf{v} \in \mathbb{R}^d$, $\texttt{softmax}(\mathbf{v})[i] := \exp(\mathbf{v}[i]) / (\sum_j \exp(\mathbf{v}[j]))$ and $\texttt{tanh}(\mathbf{v})[i] := (\exp(2\mathbf{v}[i]) - 1) / (\exp(2\mathbf{v}[i]) + 1)$.

We emphasize, `Colander` can use any function class for $\mathcal{G} : \mathcal{X} \to T^k$. Here, we chose 2-layer nets and successfully used the same across all experiments; thus, we may not need an exhaustive architecture search. Intuitively, we do not need a large network for g since $\hbar$ already performs the heavier representation learning work. As a result, simple models are preferable to avoid overfitting and to reduce training time since post-hoc methods should be fast.

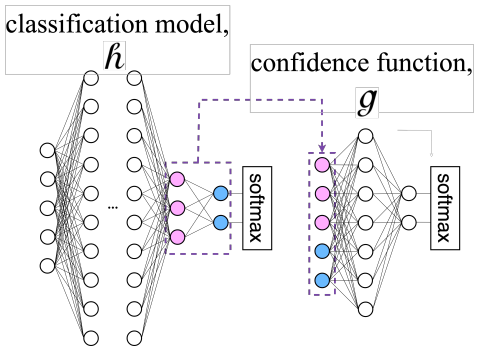


Figure 4.4: Our choice of g function.

Solving the surrogate optimization. The optimization problem (P4.3) is nonconvex. Nevertheless, it is differentiable, and we can apply gradient-based methods. We solve for g and $\mathbf{t}$ simultaneously using Adam (Kingma and Ba, 2014). Training details, including hyperparameters, are deferred to the Appendix.

4.3.3 TBAL procedure with `Colander`

We take the workflow of TBAL and plug in our method `Colander` to learn the new confidence function and threshold. We discuss the updated workflow below and place the detailed Algorithms 5 and 6 in the Appendix B.2 due to space constraints.

1. Initialization. First, select n_s points randomly from X_u and obtain human labels for them to create initial training data $D^{(1)}_{\text{train}}$. This is written as $\textsc{RandomQuery}(X_u, n_s)$ in Algorithm 5. The procedure $\textsc{RandomQuery}(X_u, n_s)$ selects n_s points randomly from X_u and obtains human labels for them to create $D^{(1)}_{\text{train}}$. The steps of this procedure are detailed in Algorithm 8 in the Appendix B.2.

2. Train classification model. After obtaining human-labeled training data $D^{(i)}_{\text{train}}$ for the current round i, the procedure $\textsc{TrainModel}(\mathcal{H}, D^{(i)}_{\text{train}})$ trains a model from model class $\mathcal{H}$ on the training data $D^{(i)}_{\text{train}}$. Any training procedure can be used here. We use methods listed in Section 4.4.1 for model training. This step outputs a model $\hat{h}_i$ trained on $D^{(i)}_{\text{train}}$. Note, this model $\hat{h}_i$ may not be highly accurate due to several factors such as the small amount of training data, the choice of $\mathcal{H}$, the training algorithm, and its hyperparameters. Indeed, TBAL does not expect the model to have high accuracy, but it aims to identify and auto-label points where the model's accuracy is at least $1 - \epsilon_a$.

3. Learn new confidence function using `Colander`. The model $\hat{h}_i$ obtained in the previous step also produces softmax scores that can be used for auto-labeling. However, as we saw earlier in Section 4.2, using these scores may lead to poor auto-labeling performance. Thus, we plug in our procedure `Colander` to learn new scores designed to auto-label as many points as possible with the current model $\hat{h}_i$ while respecting the error constraint. We first randomly splits the validation data $D^{(i)}_{\text{val}}$ into $D^{(i)}_{\text{cal}}$ and $D^{(i)}_{\text{th}}$ using procedure $\textsc{RandomSplit}(D^{(i)}_{\text{val}}, \nu)$. The part $D^{(i)}_{\text{cal}}$ has a fraction ν of the points from $D^{(i)}_{\text{val}}$. Then we consider problem P4.3 with $\hat{h}_i$ and $D^{(i)}_{\text{cal}}$. We solve it to obtain the post-hoc confidence function $\hat{g}_i$, which is expected to provide the most appropriate scores for TBAL.

We get thresholds $\hat{\mathbf{t}}'_i$ as output from `Colander`, and it is tempting to use these along with $\hat{g}_i$

for auto-labeling. However, these thresholds may violate the auto-labeling error constraint as they are obtained by solving the relaxed optimization problem. Thus, it is crucial to estimate reliable thresholds $\hat{\mathbf{t}}_i$ from the held-out data $D_{\text{th}}^{(i)}$ to ensure the auto-labeling error constraint is not violated.

4. Threshold estimation. The scores from the new confidence function $\hat{g}_i$ on $D_{\text{th}}^{(i)}$ are used to estimate auto-labeling thresholds in Algorithm 6. This procedure finds thresholds for each class separately. It first splits the points in $D_{\text{th}}^{(i)}$ according to the ground truth class into subsets $D_{\text{th}}^{(i,y)}$. Then, for each class y, it finds the auto-labeling threshold $\hat{\mathbf{t}}[y]$ by selecting the minimum threshold t such that the estimate of auto-labeling error for class y, $\widehat{\mathcal{E}}_y(\hat{g}_i, t|\hat{h}_i, D_{\text{th}}^{(i,y)})$ plus a confidence interval $\widehat{\zeta}(\widehat{\mathcal{E}}_y(\hat{g}_i, t|\hat{h}_i, D_{\text{th}}^{(i,y)}))$ estimated on points in $D_{\text{th}}^{(i,y)}$ having scores above t, is at most the given error tolerance ϵ_a. Here $\widehat{\zeta}(z) = C_1\sqrt{z(1-z)}$ for $z \in [0,1]$ and $C_1 \geq 0$ is a hyperparameter.

5. Auto-labeling. This is a simple step. We compute the scores on the remaining unlabeled data $X_u^{(i)}$ using the function $\hat{g}_i$ and any point $\mathbf{x} \in X_u^{(i)}$ having score above $\hat{\mathbf{t}}[\hat{y}]$ is assigned auto-label $\hat{y} = \hat{h}_i(\mathbf{x})$, and the points that did not meet this criterion remain unlabeled.

6. Remove auto-labeled points. The points that got auto-labeled in the previous steps are removed from the unlabeled pool. To make the validation data consistent with this unlabeled pool for the next round, the points in the validation data that fall into the auto-labeling region are also removed.

7. Get more human-labeled data. Lastly, it calls the procedure ActiveQuery$(\hat{h}_i, X_u^{(i)}, n_b)$ to select n_b points from the remaining unlabeled pool using an active learning strategy. This newly acquired human-labeled data is added to the training data $D_{\text{train}}^{(i)}$. The details of the querying strategy are in Algorithm 7, Appendix B.2. Note that the TBAL procedure is flexible to work with any choice of active querying strategy. We pick a simple strategy based on random sampling from the points where the classifier is most uncertain. To avoid confounding in TBAL with other scores, we use the softmax scores from the classifier to determine uncertainty here.

The procedure then moves to step 2 and runs the loop until there are no more unlabeled points left or it has queried the stipulated number of human labels N_t.

Dataset	Model $\hbar$	N	N_u	K	N_t	N_v	N_{hyp}	Modality	Preprocess	Dimension
MNIST	LeNet-5	70k	60k	10	500	500	500	Image	None	$1 \times 28 \times 28$
CIFAR-10	CNN	50k	40k	10	10k	8k	2k	Image	None	$3 \times 32 \times 32$
Tiny-Imagenet	MLP	110k	90k	200	10k	8k	2k	Image	CLIP	512
20 Newsgroup	MLP	11.3k	9k	20	2k	1.6k	600	Text	FlagEmb.	1,024

Table 4.1: Details of the dataset and model we used to evaluate the performance of our method and other calibration methods. For the Tiny-Imagenet and 20 Newsgroup datasets, we use CLIP and FlagEmbedding, respectively, to obtain the embeddings of these datasets and conduct auto-labeling on the embedding space. For Tiny-Imagenet, we use a 3-layer perceptron with 1,000, 500, 300 neurons on each layer as model $\hbar$; for 20 Newsgroup, we use a 3-layer perceptron with 1,000, 500, 30 neurons on each layer as model $\hbar$.

4.4 Empirical Evaluation

We validate the following claims through extensive empirical evaluation,

C1. `Colander` learns better confidence functions for auto-labeling compared to standard training and common post-hoc methods that seek to mitigate the overconfidence problem. Using it in TBAL can boost the coverage significantly while keeping the auto-labeling error low.

C2. `Colander` is independent of any particular train-time method and thus should help improve the performance when coupled with different train-time methods.

4.4.1 Baselines

We examine several train-time and post-hoc methods that improve confidence functions from the calibration and ordinal ranking perspectives. Details of these methods are in the Appendix B.3.1.

Train-time methods. We use the following methods for training the model $\hat{h}$. *Vanilla* neural networks are trained with the cross-entropy loss using stochastic gradient descent (SGD) (Amari, 1993; Bottou, 2012; Guo et al., 2017). *Squentropy* (Hui et al., 2023) adds the average square loss over the incorrect classes to the cross-entropy loss to improve the calibration and accuracy of the model. *Correctness Ranking Loss (CRL)* (Moon et al., 2020) aligns the confidence scores of the model with the ordinal rankings criterion via regularization. *FMFP* (Zhu et al., 2022) aligns confidence scores with the ordinal rankings criterion by using sharpness-aware minimization (SAM) (Foret et al., 2021) in lieu of SGD.

Post-hoc methods. We use the following methods for learning (or updating) the confidence function $\hat{g}$ after learning $\hat{h}$. *Temperature scaling* (Guo et al., 2017) is a variant of Platt scaling (Platt, 1999). It rescales the logits by a learnable scalar parameter. *Top-Label Histogram-Binning* (Gupta and Ramdas, 2022) builds on the histogram-binning method (Zadrozny and Elkan, 2002) and focuses on calibrating the scores of the predicted label assigned to unlabeled points. *Scaling-Binning* (Kumar et al., 2019) applies temperature scaling and then bins the confidence function values. *Dirichlet Calibration* (Kull et al., 2019) models the distribution of predicted probability vectors separately on instances of each class and assumes Dirichlet class conditional distributions. *Adaptive Temperature Scaling* (Joy et al., 2023) builds on top of temperature scaling and considers that different samples contribute to the calibration error differently. Each train-time method is piped with a post-hoc method, yielding a total of $4 \times 6 = 24$ methods.

4.4.2 Datasets and models

We evaluate the performance of auto-labeling on four datasets. Each is paired with a model for auto-labeling: *MNIST* LeCun (1998) is a hand-written digits dataset. We use the LeNet LeCun et al. (1998) for auto-labeling. *CIFAR-10* Krizhevsky et al. (2009a) is an image dataset with 10 classes. We use a CNN with approximately 5.8M parameters Hussain (2021) for auto-labeling. *Tiny-ImageNet* Le and Yang (2015b) is an image dataset comprising 100K images across 200 classes. We use CLIP Radford et al. (2021) to derive embeddings for the images in the dataset and use an MLP model. *20 Newsgroups* Mitchell (1999) is a natural language dataset comprising around 18K news posts across 20 topics. We use the FlagEmbedding Xiao et al. (2023) to obtain text embeddings and use an MLP model.

4.4.3 Hyperparameter search and evaluation

The complexity of the TBAL workflow and the lack of labeled data make hyperparameter search and evaluation challenging. Similar challenges have been observed in active learning (Lowell et al., 2019). We discuss our practical approach and defer the details to Appendix B.3.3 and

code[1].

Hyperparameter search. We run only the first round of TBAL with each method using a hyperparameter combination 5 times and measure the mean auto-labeling error and mean coverage on D_{hyp}, which represents a small part of the held-out human-labeled data. We pick the combination that yields the lowest average auto-labeling error while maximizing the coverage. We first find the best hyperparameters for each train-time method, fix those, and then search the hyperparameters for the post-hoc methods. Note that the best hyperparameter for a post-hoc method depends on the training-time method that it pipes to. The hyperparameter search spaces are in the Appendix B.3; and the selected values used for each setting are in the supplementary material.

Performance evaluation. After fixing the hyperparameters, we run TBAL with each combination of train-time and post-hoc method on full X_u of size N, with a fixed budget of N_t labeled training samples and N_v validation samples. The details of these values for each dataset are in Table 4.1 in Appendix B.3. Here, we know the ground truth labels for the points in X_u, so we measure the auto-labeling error and coverage as defined in (4.1) and report them in Table 4.2.

4.4.4 Results and discussion

Our findings, shown in Table 4.2, are:

C1: `Colander` improves TBAL performance. Our approach aims to optimize the confidence function to maximize coverage while minimizing errors. When applied to TBAL, we expect it to yield substantial coverage enhancement and error reduction compared to vanilla training and softmax scores. Indeed, the results in Table 4.2 corresponding to the vanilla training match our expectations. We see *across all data settings*, our method achieves *significantly higher coverage* while keeping auto-labeling error below the tolerance level of 5%. The improvements are even more pronounced when the datasets are more complex than MNIST. Also consistent with our expectation and observations in Figure 4.2(b), the post-hoc calibration methods improve the

[1] https://github.com/harit7/TBAL-Colander-NeurIPS-24

coverage over using softmax scores but at the cost of slightly higher error. While they are reasonable choices to apply in the TBAL pipeline, they fall short of maximally improving TBAL performance due to the misalignment of goals.

C2: `Colander` is compatible with and improves over other train-time methods. Our method is compatible with various choices of train-time methods, and if a train-time method (Squentropy here) provides a *better* model relative to another train-time method (e.g., Vanilla), then our method exploits this gain and pushes the performance even further. Across different train-time methods, we do not see significant differences in the performance, except for Squentropy. Using Squentropy with softmax improves the coverage by as high as 6-7% while dropping the auto-labeling error in contrast to using softmax scores obtained with other train-time methods for the Tiny-ImageNet setting. This is unexpected: Squentropy adds the average square loss over the incorrect classes as a regularizer, and it has offered better accuracy and calibration compared to training with cross-entropy loss.

Train-time methods designed for ordinal ranking objective perform poorly in auto-labeling. CRL and FMFP are state-of-the-art methods designed to produce scores aligned with the ordinal ranking criteria. Ideally, if the scores satisfy this criterion, TBAL's performance would improve. However, we do not see any significant difference from the Vanilla method. Similar to the other baselines, their evaluation is focused on models trained on large amounts of data. But, in TBAL, we have less data for training. The training error goes to zero after some rounds, and no information is left for the CRL loss to distinguish between correct and incorrect predictions (i.e., count SGD mistakes). On the other hand, FMFP is based on a hypothesis that training models using Sharpness Aware Minimizer (SAM) could lead to scores satisfying the ordinal ranking criteria. However, this phenomenon is still not well understood, especially in settings like ours with limited training data.

4.5 Related Work

Data labeling. We briefly discuss prominent methods for labeling. Crowdsourcing Raykar et al. (2010); Sorokin and Forsyth (2008) uses a crowd of non-experts to complete a set of labeling tasks. Works in this domain focus on denoising the obtained information, modeling label errors, and designing effective labeling tasks Gomes et al. (2011); Karger et al. (2011); Mazumdar and Saha (2017); Vinayak et al. (2014); Vinayak and Hassibi (2016); Vinayak et al. (2017); Chen et al. (2023a). Weak supervision (WS), in contrast, emphasizes labeling through multiple inexpensive but noisy sources, not necessarily human Ratner et al. (2016); Shin et al. (2022b); Vishwakarma and Sala (2022); Huang et al. (2023b); Shin et al. (2023); Yu et al. (2021); Zhu et al. (2023). Works such as Ratner et al. (2016); Fu et al. (2020) concentrate on binary or multi-class labeling, while Shin et al. (2022b); Vishwakarma and Sala (2022) extend WS to structured prediction tasks.

Auto-labeling occupies an intermediate position between weak supervision and crowdsourcing in terms of human dependency. It aims to minimize costs to obtain human labels while generating high-quality labeled data using a specific model. Qiu et al. (2023) use a TBAL-like algorithm and explore the cost of training for auto-labeling with large-scale model classes. Recent work Vishwakarma et al. (2023) theoretically analyzes the sample complexity of validation data required to guarantee the quality of auto-labeled data.

Overconfidence and calibration. The issue of overconfidence Szegedy et al. (2014); Nguyen et al. (2015a); Hein et al. (2018); Bai et al. (2021) is detrimental in several applications (such as robustness to out-of-distribution points Yang et al. (2022a); Vishwakarma et al. (2024c)), including ours. Many solutions have emerged to mitigate the overconfidence and miscalibration problems. Gawlikowski et al. (2021b) provide a comprehensive survey on uncertainty quantification and calibration techniques for neural networks. Guo et al. (2017) evaluated a variety of solutions ranging from the choice of network architecture, model capacity, weight decay regularization Krogh and Hertz (1991), histogram-binning and isotonic regression Zadrozny and Elkan (2001, 2002) and temperature scaling Platt (1999); Niculescu-Mizil and Caruana (2005) which they found to be the most promising solution. The solutions fall into two broad categories: train-time and

post-hoc. Train-time solutions modify the loss function, include additional regularization terms, or use different training procedures Kumar et al. (2018); Müller et al. (2019); Mukhoti et al. (2020); Hui et al. (2023). On the other hand, post-hoc methods such as top-label histogram-binning Gupta and Ramdas (2021), scaling binning Kumar et al. (2019), Dirichlet calibration Kull et al. (2019) calibrate the scores directly or learn a model that corrects miscalibrated confidence scores.

Beyond calibration. While calibration aims to match the confidence scores with a probability of correctness, it is not the precise solution to the overconfidence problem in many applications, including our setting. The desirable criteria for scores for TBAL are closely related to the ordinal ranking criterion Hendrycks and Gimpel (2017a). To get such scores, Corbière et al. (2019) add a module in the net for failure prediction, Zhu et al. (2022) switch to sharpness aware minimization Foret et al. (2021) to learn the model; CRL Moon et al. (2020) regularizes the loss.

4.6 Conclusion

In this chapter, we studied issues with confidence scoring functions used in threshold-based auto-labeling (TBAL). We showed that the commonly used confidence functions and calibration methods can often be a bottleneck, leading to poor performance. We proposed `Colander` to learn confidence functions that are aligned with the TBAL objective. We evaluated our method extensively against common baselines on several real-world datasets and found that it improves the performance of TBAL significantly in comparison to several common choices of confidence function. Our method is compatible with several choices of methods used for training the classifier in TBAL, and using it in conjunction with them improves TBAL performance further. A limitation of `Colander` is that, similar to other post-hoc methods, it also requires validation data to learn the confidence function. Reducing (or eliminating) this dependence on validation data could be an interesting future work.

Train-time	Post-hoc	MNIST		CIFAR-10		20 Newsgroups		Tiny-ImageNet	
		Err (↓)	**Cov (↑)**	**Err (↓)**	**Cov (↑)**	**Err (↓)**	**Cov (↑)**	**Err (↓)**	**Cov (↑)**
Vanilla	Softmax	**4.1±0.7**	85.0±2.5	4.8±0.2	14.0±2.1	6.0±0.6	48.2±1.6	11.1±0.3	32.6±0.5
	TS	7.8±0.6	94.2±0.5	7.3±0.3	23.2±0.7	9.7±0.6	60.7±2.3	16.3±0.5	37.4±1.5
	Dirichlet	7.9±0.7	93.2±2.2	7.7±0.5	22.4±1.2	9.4±0.9	59.4±1.8	17.1±0.4	33.3±2.0
	SB	6.7±0.5	92.6±1.5	6.1±0.4	18.6±1.1	8.1±0.6	58.1±1.8	15.7±0.6	35.4±1.2
	Top-HB	7.4±1.4	93.1±3.6	6.0±0.7	15.6±1.9	9.2±1.0	59.0±2.0	16.6±0.5	37.6±2.2
	AdaTS	7.5±0.9	92.8±2.0	8.6±0.6	16.9±1.0	9.6±1.1	61.8±3.3	15.9±0.7	36.7±1.9
	Ours	4.2±1.5	**95.6±1.4**	**3.0±0.2**	**78.5±0.2**	**2.5±1.1**	**80.6±0.7**	**1.4±2.1**	**59.2±0.8**
CRL	Softmax	4.7±0.4	86.0±4.5	5.2±0.3	15.9±0.8	5.8±0.5	48.3±0.3	10.4±0.4	32.5±0.6
	TS	8.0±0.8	94.8±0.8	6.8±0.8	20.3±1.1	9.5±1.0	61.7±1.6	15.8±0.6	37.4±1.7
	Dirichlet	8.6±0.6	93.1±1.6	7.7±0.2	20.9±1.1	8.7±0.9	58.0±1.4	16.3±0.4	33.1±1.9
	SB	7.4±0.8	93.1±2.7	5.9±0.9	17.9±1.5	8.9±1.1	57.9±3.9	15.0±0.4	35.5±1.2
	Top-HB	7.7±0.8	94.1±1.5	4.4±0.5	12.3±0.4	8.8±1.0	58.8±2.7	16.5±0.5	38.9±1.6
	AdaTS	7.8±0.7	94.3±1.2	8.8±0.4	17.1±1.2	9.1±0.8	60.8±1.9	16.2±0.4	38.9±1.2
	Ours	**4.5±1.4**	**95.6±1.3**	**2.2±0.6**	**77.9±0.2**	**1.8±1.2**	**81.3±0.5**	**2.8±2.1**	**61.2±1.4**
FMFP	Softmax	4.8±0.8	84.2±4.1	4.9±0.4	15.6±1.7	5.4±0.7	45.4±1.9	10.5±0.3	32.4±1.4
	TS	8.0±0.6	95.3±1.6	6.5±0.3	21.0±1.5	9.5±0.5	57.7±2.2	16.2±1.1	37.7±1.8
	Dirichlet	8.2±1.3	94.0±2.2	6.9±0.4	21.7±1.2	8.9±1.0	56.6±2.4	17.4±0.8	33.0±1.8
	SB	7.2±1.1	93.1±2.3	6.1±0.5	19.5±1.0	8.6±0.4	55.8±1.3	15.5±0.6	36.1±0.5
	Top-HB	7.1±0.6	93.3±4.9	5.2±0.5	14.2±2.4	9.0±0.7	57.9±2.4	16.2±0.4	37.4±1.1
	AdaTS	7.6±0.4	94.1±1.0	7.2±0.7	27.5±1.5	8.7±0.9	56.7±2.7	16.3±0.6	37.6±1.7
	Ours	**4.6±0.8**	**95.7±0.2**	**3.0±0.4**	**77.4±0.2**	**2.5±0.9**	**80.8±0.6**	**1.8±2.0**	**60.8±1.4**
Squentropy	Softmax	**3.7±1.0**	88.2±3.9	5.2±0.5	21.2±1.8	4.6±0.4	52.0±1.2	7.8±0.3	36.2±0.8
	TS	6.2±1.1	95.6±0.9	6.9±0.6	28.2±2.5	8.3±0.6	66.6±1.4	13.3±0.1	44.9±1.0
	Dirichlet	6.5±1.2	95.9±0.8	7.3±0.3	29.4±1.1	7.8±0.6	64.0±1.3	14.1±0.3	42.5±0.7
	SB	6.0±0.8	95.3±1.2	6.2±0.4	23.8±1.9	7.8±0.7	63.0±2.9	13.0±0.5	45.2±2.0
	Top-HB	5.3±0.4	96.4±0.9	4.3±0.5	15.8±1.4	8.2±0.8	66.5±2.2	13.7±0.1	45.9±1.4
	AdaTS	6.6±1.0	95.7±1.0	7.6±0.3	22.6±1.2	7.4±0.6	64.7±2.6	14.0±0.3	46.1±0.7
	Ours	4.1±0.8	**97.2±0.5**	**2.3±0.5**	**79.0±0.3**	**3.3±0.8**	**82.9±0.4**	**0.6±0.2**	**66.5±0.7**

Table 4.2: In every round, the error was enforced to be below 5%; 'TS' stands for Temperature Scaling, 'SB' stands for Scaling Binning, 'Top-HB' stands for Top-Label Histogram Binning. 'AdaTS' stands for Adaptive Temperature Scaling. The column Err stands for auto-labeling error, and Cov stands for coverage. Each cell value is the mean ± std. deviation on 5 repeated runs with different random seeds.

Chapter 5

Confidence Scores and Thresholds in Pseudolabeling-based SSL

Modern semi-supervised learning (SSL) methods rely on pseudolabeling and consistency regularization. Pseudolabeling is typically performed by comparing the model's confidence scores and a predefined threshold. While several heuristics have been proposed to improve threshold selection, the underlying issues of overconfidence and miscalibration in confidence scores remain largely unaddressed, leading to inaccurate pseudolabels, degraded test accuracy, and prolonged training. We take a first-principles approach to learn confidence scores and thresholds with an explicit knob for error. This flexible framework addresses the fundamental question of optimal scores and threshold selection in pseudolabeling. Moreover, it gives practitioners a principled way to control the quality and quantity of pseudolabels. Such control is vital in SSL, where balancing pseudolabel quality and quantity directly affects model performance and training efficiency. Our experiments show that, by integrating this framework with modern SSL methods, we achieve significant improvements in accuracy and training efficiency. In addition, we provide novel insights on the trade-offs between the choices of the error parameter and the end model's performance.

5.1 Introduction

The lack of high-quality labeled data is a major bottleneck in training high-accuracy models. The semi-supervised learning (SSL) paradigm tackles this problem by leveraging abundant unlabeled data alongside a limited set of labeled examples (Chapelle et al., 2006; Zhu, 2005; van Engelen and Hoos, 2019). While SSL dates back decades and includes a wide variety of approaches, modern SSL methods frequently rely on a pair of ideas: *self-training or pseudolabeling* – where model generated labels are assigned to unlabeled data for further training (McLachlan, 1975; Amini et al., 2023; Rosenberg et al., 2005; Lee et al., 2013; Rizve et al., 2021) – and *consistency regularization* to enforce stability in predictions across perturbed inputs (Laine and Aila, 2017; Bachman et al., 2014; Sajjadi et al., 2016; Fan et al., 2021; Kukačka et al., 2017). SSL techniques with these ideas have performed strongly on several benchmark datasets.

While pseudolabeling is a powerful technique, its effectiveness hinges on a fundamental question: *which points should be labeled using the model's predictions?* Since pseudolabels are derived from a model being trained, they can be highly unreliable. A naive approach that assigns pseudolabels too liberally risks injecting noisy labels into training, amplifying the model's existing errors – a phenomenon known as confirmation bias. Conversely, an overly conservative approach that selects only the correct predictions severely limits the amount of useful training data. Both extremes can degrade SSL performance, leading to either poor generalization or slow convergence. To fully harness the potential of pseudolabeling, we need a principled approach for pseudolabeling with explicit control on these trade-offs.

A widely used strategy pseudolabels points on which the model's confidence score exceeds a threshold. This approach provides a simple mechanism for selecting points while controlling the quality and quantity of pseudolabels via thresholds. However, the prior works based on this approach suffer from two key limitations that restrict their effectiveness. First, thresholding techniques are often heuristic-driven, lacking a precise control for a target error level (Sohn et al., 2020; Zhang et al., 2021b; Wang et al., 2023; Xu et al., 2021; Chen et al., 2023b; Li et al., 2023). Second, commonly used scores, such as the model's softmax outputs, tend to be unreliable. Recent

studies in this vein (Loh et al., 2022; Mishra et al., 2024; Rizve et al., 2021) have highlighted issues of overconfidence and miscalibration in these scores, leading to inefficiencies in pseudolabeling.

In addition to these problems with the choices of scores and thresholds, an equally important question remains: how much error should be allowed in pseudolabeling? As discussed earlier, a very low tolerance may pseudolabel too few points, and conversely, a high tolerance may allow for large errors, hurting the efficiency of pseudolabeling in both cases. To solve these issues, we seek *a principled solution to get confidence scores and thresholds for pseudolabeling at a given target level of error tolerance.*

We take a first principles approach to formalize the pseudolabeling objective: maximize the number of pseudolabeled points while adhering to the given error tolerance. We formulate this objective as an optimization problem over a flexible space of confidence functions and thresholds. By solving this optimization problem, we obtain confidence scores tailored to our objective of pseudolabeling. To ensure the pseudolabeling error constraint is strictly followed, we use a separate procedure to estimate thresholds on these scores using part of the validation data. Pseudolabeling with these scores and thresholds ensures we are pseudolabeling a maximal set of points that can be pseudolabeled at the given error tolerance level.

We integrate this approach into popular pseudolabeling-based SSL methods, providing two benefits. First, it provides a principled way to derive confidence scores and thresholds for any given error tolerance. Second, by enabling more precise pseudolabeling, it improves the utilization of the unlabeled data and is expected to yield an end model with higher test accuracy compared to the ad-hoc choices of scores and thresholds.

Our main contributions are summarized as follows,

1. Our work settles the question of the right choices of confidence scores and thresholds in pseudolabeling-based SSL methods by introducing a framework for learning confidence scores and thresholds. Departing from heuristic-driven or ad-hoc and unreliable choices of scores and thresholds, this framework provides principled choices of scores and thresholds for pseudolabeling with any target error tolerance.

2. We show how this framework for learning confidence scores and thresholds can work in concert with popular SSL methods such as Fixmatch (Sohn et al., 2020), Freematch (Wang et al., 2023), etc., and conduct an extensive empirical evaluation demonstrating that by pseudolabeling using confidence scores and thresholds learned from our method can yield significant improvements in the test accuracy.

3. Leveraging our framework's ability to pseudolabel at any target error level, we study the impact of varying pseudolabeling error levels, from fixed to dynamic tolerance throughout training. Our results confirm the intuition that lower pseudolabeling errors lead to better end models compared to higher errors. Moreover, among dynamic schedules, it is better to use a decreasing schedule of error tolerances.

5.2 Background and Problem Setup

We begin with notation, then provide useful background and a statement of our goal.

Notation. Consider a feature space $\mathcal{X}$ and label space $\mathcal{Y} = \{1, \ldots, k\}$ in a k-class classification task. As usual in semi-supervised learning, we have access to a set $X_u = \{\mathbf{x}_u\}_{u=1}^{n_u}$ of unlabeled data drawn from the distribution P_x over $\mathcal{X}$. We also have access to $D_l = \{(\mathbf{x}_l, y_l)\}_{l=1}^{N_l}$, a set of labeled data points drawn from the joint distribution P_{xy}, with $N_l \ll n_u$. Let $\hbar : \mathcal{X} \to \mathcal{Y}$ denote a model and $g : \mathcal{X} \to T^k \subseteq \mathbb{R}^k$ be an associated confidence function giving a score $g(\mathbf{x})$ indicating the confidence of $\hbar$ on its prediction for any data point $\mathbf{x}$. For any $\mathbf{x}$ the hard label prediction is $\widehat{y} := \hbar(\mathbf{x})$. When the prediction $\widehat{y}$ is used as a pseudolabel, we denote it as $\tilde{y}$. In general, for a vector $\mathbf{v} \in \mathbb{R}^d$, $\mathbf{v}[i]$ denotes its $i-$th component. The vector $\mathbf{t}$ denotes thresholds over the scores k-classes, and $\mathbf{t}[y]$ is its $y-$th entry, i.e., the threshold for class y.

5.2.1 Pseudolabeling-based Semi-Supervised Learning

Given, as above, a large collection of unlabeled data X_u and a small set of labeled points D_l, inductive semi-supervised learning (SSL) seeks to learn a classifier $\widehat{h}_{\text{ssl}}$ from the model class $\mathcal{H}$. The promise of SSL is that by effectively using X_u in the learning process, it can learn a better

classifier than its supervised counterpart, which learns only from D_l.

In many recent pseudolabeling-based SSL techniques, in each iteration of training, a batch of labeled and unlabeled data is obtained, then the sum of the losses $\widehat{\mathcal{L}} = \widehat{\mathcal{L}}_s + \lambda_u \widehat{\mathcal{L}}_u + \lambda_r \widehat{\mathcal{L}}_r$ is minimized with respect to the model h. Here $\widehat{\mathcal{L}}_s$ is the supervised loss, $\widehat{\mathcal{L}}_u$ unsupervised loss, and $\widehat{\mathcal{L}}_r$ is (the sum of) regularization term(s). The constants λ_u, λ_r are hyperparameters controlling the relative importance of the corresponding terms.

Supervised loss. Given a batch of labeled data, D_l^b the supervised loss is computed as follows, $\widehat{\mathcal{L}}_s(h \mid D_l^b) = \frac{1}{|D_l^b|} \sum_{(\mathbf{x},y)\in D_l^b} H(y, h, \mathbf{x})$. Here $H(y, h, \mathbf{x})$ is the standard cross-entropy loss between the 1-hot representation of y and the softmax output of h on input $\mathbf{x}$.

Unsupervised loss and consistency regularization. For the unlabeled batch X_u^b, pseudolabels $\widetilde{y} = h(\mathbf{x})$ are computed for each $\mathbf{x} \in X_u^b$. Then, a pseudolabeling mask, $S(\mathbf{x}, g, \mathbf{t} \mid h) = \mathbb{1}(g(\mathbf{x})[\widetilde{y}] \geq \mathbf{t}[\widetilde{y}])$ is computed. It is 1 for points having confidence score $g(\mathbf{x})[\widetilde{y}]$, bigger than predetermined threshold $\mathbf{t}[\widetilde{y}]$, corresponding to the predicted class $\tilde{y}$. Recent methods couple this loss and consistency regularization together by doing pseudolabeling on weakly augmented data using weak transform $\omega : \mathcal{X} \mapsto \mathcal{X}$ and then defining the cross-entropy loss on the strongly augmented data using strong transformation $\Omega : \mathcal{X} \mapsto \mathcal{X}$. The loss is,

$$\widehat{\mathcal{L}}_u := \frac{1}{|\widetilde{D}_u^b|} \sum_{(x,\tilde{y})\in\tilde{D}_u^b} S(\omega(\mathbf{x}), g, \mathbf{t}|h) \cdot H(\tilde{y}, h, \Omega(\mathbf{x})).$$

Regularization. A regularization term (or a sum of multiple regularizers) is often included along with the above two losses to encourage desired behavior(s). For instance, Freematch (Wang et al., 2023) adds a self-adaptive class fairness regularizer to encourage diverse predictions during the initial training phase. Similarly, a regularizer is introduced in (Mishra et al., 2024) to encourage calibration in the model's confidence scores. Including such regularizers has been fruitful in pseudolabeling-based SSL.

5.2.2 Quality and Quantity of Pseudolabels

Given a classifier h, the quality and quantity of the pseudolabels w.r.t. to score function g and thresholds $\mathbf{t}$, are:

Pseudolabeling coverage (quantity). Given a set of points X, the pseudolabeling coverage is the fraction of points that are pseudolabeled using h, g and $\mathbf{t}$. This measurement captures the quantity of pseudolabels and is defined as

$$\widehat{\mathscr{P}}(g, \mathbf{t} \mid h, X) := \frac{1}{|X|} \sum_{(\mathbf{x}) \in X} S(\mathbf{x}, g, \mathbf{t} \mid h) \tag{5.1}$$

$$\mathscr{P}(g, \mathbf{t} \mid h) := \mathbb{E}_{\mathbf{x}}[S(\mathbf{x}, g, \mathbf{t} \mid h)]. \tag{5.2}$$

Pseudolabeling error (quality). This is the fraction of pseudolabeled points that got incorrect labels. This metric captures the quality of pseudolabels:

$$\widehat{\mathscr{E}}(g, \mathbf{t} \mid h, D) := \frac{\sum_{(\mathbf{x}, y, \tilde{y}) \in D} S(\mathbf{x}, g, \mathbf{t} \mid h) \cdot \mathbb{1}(h(\mathbf{x}) \neq y)}{\sum_{(\mathbf{x}, y, \tilde{y}) \in D} S(\mathbf{x}, g, \mathbf{t} \mid h)}, \tag{5.3}$$

$$\mathscr{E}(g, \mathbf{t} \mid h) = \frac{\mathbb{E}_{\mathbf{x}}[S(\mathbf{x}, g, \mathbf{t} \mid h) \cdot \mathbb{1}(h(\mathbf{x}) \neq y)]}{\mathscr{P}(g, \mathbf{t}|h)}. \tag{5.4}$$

5.2.3 Our Goals

Pseudolabeling-based SSL aims to learn a classifier $\widehat{h}_{\text{ssl}}$ that generalizes well on the unseen data, i.e., has high test accuracy. This is typically achieved by pseudolabeling points using confidence scores and thresholds and incorporating them into training. However, existing choices of scores and thresholding strategies are often ad-hoc and unreliable, limiting their effectiveness. Departing from these unreliable approaches, our goal is to:

(i) Design principled solutions for confidence scores and thresholding to maximize the number of

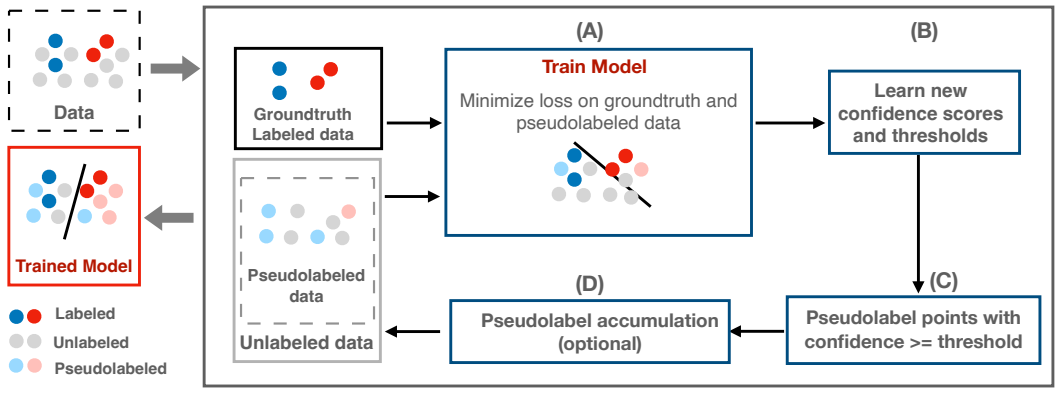


Figure 5.1: Workflow of pseudolabeling-based SSL with `PabLO` (A): Train model with standard supervised loss, consistency regularization, and other regularizers (B): Learn new confidence scores and thresholds (C): Pseudolabel points with scores greater than the estimated thresholds (D): An optional pseudolabeling accumulation to use previous pseudolabels for points that are not pseudolabeled in the current round. Note that the pseudolabels can be noisy (incorrect). The training and pseudolabeling loop continues until a pre-specified number of iterations. In the end, it outputs a model $\widehat{h}_{\text{ssl}}$ that is expected to have higher test accuracy than the model trained only on the given groundtruth training data. Note that in the end, there might be points left unlabeled, and the pseudolabels might be noisy.

pseudolabeled points while ensuring the pseudolabeling error is at most ϵ.

(ii) Incorporate these in the existing pseudolabeling-based SSL methods and assess whether this gives a better end model $\widehat{h}_{\text{ssl}}$.

(iii) Study the sensitivity of the SSL pipeline to pseudolabeling errors by leveraging the ability of our approach to explicitly ensure the pseudolabeling error remains below ϵ.

5.3 Methodology

In this section, we discuss our principled framework to learn scores and thresholds with explicit control of the pseudolabeling errors and use them in pseudolabeling-based SSL.

5.3.1 Pseudolabeling Optimization Framework

Given the current model $\widehat{h}_i$ in the ith iteration, can we obtain confidence scores and thresholds using which we can identify a maximal set of points that can be pseudolabeled using $\widehat{h}_i$ with at

most ϵ error? We begin with a theoretical formulation to learn such scores and thresholds, and then introduce its practical version.

Theoretical framework. Instead of improving calibration or heuristics for thresholding, we propose to express the objective of pseudolabeling as an optimization problem over the space of confidence functions and thresholds. The objective is to maximize the quantity, i.e., the pseudolabeling coverage (eq. (5.2)) while keeping the pseudolabeling error (eq. (5.4)) below a tolerance level $\epsilon \in (0, 1)$. More specifically, given the classifier $\widehat{h}_i$ in any iteration i of SSL,

$$g_i^\star, \mathbf{t}_i^* \in \underset{g \in \mathcal{G}, \mathbf{t} \in T^k}{\arg\max} \mathscr{P}(g, \mathbf{t} \mid \widehat{h}_i) \text{ s.t. } \mathscr{E}(g, \mathbf{t} \mid \widehat{h}_i) \leq \epsilon, \tag{5.5}$$

are the optimal confidence functions and thresholds for pseudolabeling using $\widehat{h}_i$'s predictions such that the pseudolabeling error is bounded by ϵ. This frees us from arbitrary choices of confidence scores, calibration techniques, and thresholding heuristics. Instead, solving the optimization problem over a flexible enough space will subsume specific strategies. Next, we discuss how to make this framework tractable to obtain scores and thresholds in practice.

Practical version. The optimization problem discussed earlier involves population-level quantities that are usually not accessible in practice. Thus, we have to use their finite sample estimates and smooth variations to make the optimization problem tractable. Specifically, the coverage and error are estimated using a small amount of held-out labeled data (called calibration data D_{cal}) curated from the validation data. Then differentiable surrogates for the 0-1 variables are introduced. Let $\sigma(\alpha, z) := 1/(1 + \exp(-\alpha z))$ denote the sigmoid function on $\mathbb{R}$ with scale parameter $\alpha \in \mathbb{R}$. The surrogates are as follows,

$$\widetilde{\mathscr{P}}(g, \mathbf{t} \mid \hbar, D_{\text{cal}}) := \frac{1}{|D_{\text{cal}}|} \sum_{(\mathbf{x}, y, \widetilde{y}) \in D_{\text{cal}}} \sigma\big(\alpha, g(\mathbf{x})[\widetilde{y}] - \mathbf{t}[\widetilde{y}]\big), \tag{5.6}$$

$$\widetilde{\mathscr{E}}(g, \mathbf{t} \mid \hbar, D_{\text{cal}}) := \frac{\sum_{(\mathbf{x}, y, \widetilde{y}) \in D_{\text{cal}}} \mathbb{1}\big(y \neq \widetilde{y}\big)\, \sigma\big(\alpha, g(\mathbf{x})[\widetilde{y}] - \mathbf{t}[\widetilde{y}]\big)}{\sum_{(\mathbf{x}, y, \widetilde{y}) \in D_{\text{cal}}} \sigma\big(\alpha, g(\mathbf{x})[\widetilde{y}] - \mathbf{t}[\widetilde{y}]\big)} \tag{5.7}$$

Dataset	Backbone Model h	k	n_u	N_{val}	N_{test}	N_l	N_{cal}	N_{th}	Augmentation
CIFAR-10	WRN-28-2	10	50K	6K	4K	250	1K	1K	Weak, Strong
CIFAR-100	WRN-28-2	100	50K	6K	4K	2500	3K	3K	Weak, Strong
SVHN	WRN-28-2	10	604,388	15,620	10,412	250	3K	3K	Weak, Strong

Table 5.1: Details of the dataset we use in our experiments. k is the number of classes. N_l is the number of labeled data points used for training the backbone model h. n_u is the number of unlabelled data points used for consistency regularization and pseudolabeling for all the methods. N_{val} is the number of points used for model selection in all methods. N_{test} is the number of test data points. N_{cal} is the number of points used for learning the g function. N_{th} is the number of data points used for threshold estimation.

Using these surrogates, the following practical optimization problem is obtained. It is also converted into an unconstrained formulation by introducing the penalty term $\lambda \in \mathbb{R}^+$ controlling the relative importance of the error and coverage.

$$\widehat{g}_i, \widehat{\mathbf{t}}_i' \in \underset{g \in \mathcal{G}, \mathbf{t} \in T^k}{\arg\min} \; -\widetilde{\mathscr{P}}(g, \mathbf{t} \mid \widehat{h}_i, D_{\text{cal}}) \; + \; \lambda \, \widetilde{\mathscr{E}}(g, \mathbf{t} \mid \widehat{h}_i, D_{\text{cal}}) \tag{P5.1}$$

We use 2-layer neural nets as a choice of $\mathcal{G}$ and $T^k = [0,1]^k$. The optimization problem (P5.1) is non-convex but differentiable, and we solve it using Stochastic Gradient Descent (SGD). See Appendix C.3 for more details on our choice of $\mathcal{G}$ and training details and hyperparameters.

5.3.2 Threshold Estimation

While we can obtain both the confidence scores and thresholds by solving (P5.1), we propose to estimate thresholds separately on a held-out part of the validation data to avoid potential generalization issues due to learning them simultaneously from the same data D_{cal} and ensure that the pseudolabeling error constraint is strictly adhered to.

When dealing with datasets containing many classes, there may not be enough samples per class to estimate reliable thresholds. Thus, to accommodate these possibilities, we consider two variations of the threshold estimation procedure: (i) estimate a common (joint) threshold for all classes and (ii) estimate separate (classwise) thresholds for each class. The procedures are outlined in Algorithm 10 and Algorithm 9 in the Appendix. We discuss them briefly here.

The procedure takes in a confidence function $\widehat{g}_i$ and part of the held-out validation data referred to as D_{th}. The idea is to estimate errors at several thresholds on this data and then pick the smallest threshold. This can be done separately for each class to obtain classwise thresholds or a common threshold for all classes. We discuss classwise thresholding here. First the data D_{th} is partitioned into k subsets $D_{\text{th}}^{(y)}$ corresponding to each class $y \in \mathcal{Y}$. Here, we slightly abuse notation: instead of $\mathbf{t} \in T^k$, we use $t \in T$ in the estimate of pseudolabeling error at threshold t for class y. To obtain threshold $\widetilde{\mathbf{t}}[y]$ for class y, the procedure finds the smallest $t \in T$ such that $\widehat{\mathcal{E}}(\widehat{g}_i, t \mid h, D_{\text{th}}^{(y)}) + C_1 \widehat{\sigma}(\widehat{\mathcal{E}}) \leq \epsilon$. Here C_1 is a constant (we use $C_1 = 0.25$) and $\widehat{\sigma}(z) = \sqrt{z \cdot (1-z)}$ and $\widehat{\mathcal{E}}$ is used for brevity in place of $\widehat{\mathcal{E}}(\widetilde{g}_i, t \mid h, D_{\text{th}}^{(y)})$. The same process is used for joint threshold estimation, where a single threshold t is estimated using the entire D_{th} and the same t is used for all classes. Using the thresholds found using these procedures ensures pseudolabeling error remains below (or close to) the tolerance level ϵ.

Remarks. Departing from fixed thresholds as in (Sohn et al., 2020), prior works have proposed adaptive and class-wise heuristic thresholding schemes based on the model's learning status, such as in (Djurisic et al., 2023; Zhang et al., 2021b; Wang et al., 2023) and others. In contrast, our approach is a principled way to estimate adaptive and class-wise pseudolabeling thresholds while providing strict control over the quality of pseudolabels. Similar procedures have been used in the context of creating reliable datasets and are backed by theoretical guarantees for the quality of pseudolabels produced (Vishwakarma et al., 2023).

5.3.3 Pseudolabeling and Accumulation

In the usual pseudolabeling-based SSL setups, the pseudolabels inferred by the model for a mini-batch are discarded after each iteration. Moreover, it is not guaranteed that a previously pseudolabeled point will get pseudolabeled in the current iteration as well. Given that the quality of pseudolabels is high, it is appealing to reuse the past pseudolabel for a point that did not get pseudolabeled in the current iteration. We propose to do so for techniques where the quality of pseudolabels is assured. We refer to this as "pseudolabel accumulation".

Mathematically, let $\widetilde{y}_j^{(i-1)} = \widetilde{Y}_u^{(i-1)}[j]$ and $\widetilde{y}_j^{(i)} = \widetilde{Y}_u^{(i)}[j]$ be the previous and current (fresh)

pseudolabels for jth unlabeled point. Let the corresponding masks (indicating whether the score is above the threshold) for these psuedolabels be $S_u^{(i-1)}[j]$ and $S_u^{(i)}[j] = \mathbb{1}(\widehat{g}_i(\mathbf{x}_j)[\widetilde{y}_j^{(i)}] \geq \widehat{\mathbf{t}}_i[\widetilde{y}_j^{(i)}])$. Then with accumulation,

$$\widetilde{Y}_u^{(i)}[j] \leftarrow S_u^{(i)}[j]\widetilde{Y}_u^{(i)}[j] + (1 - S_u^{(i)}[j])\widetilde{Y}_u^{(i-1)}[j],$$
$$S_u^{(i)}[j] \leftarrow S_u^{(i)}[j] \vee S_u^{(i-1)}[j].$$

Here $\vee$ is the boolean `or` operation, and the steps are executed in the order. In words, if the point is pseudolabeled in the current iteration (i.e., its current mask is 1), then it will use the current pseudolabel; otherwise, if the point was pseudolabeled in earlier iteration(s) it will use the pseudolabel from that iteration and mark the point as pseudolabeled. In case the point is not pseudolabeled in this iteration or any other iteration in the past, it will remain unlabeled. While it is appealing to use this trick, its use is only warranted in settings ensuring high-quality pseudolabels. We try to understand the consequences of the inclusion and exclusion of this trick in pseudolabeling-based SSL via experiments discussed in the next section.

We put together the steps for learning scores, thresholds, and performing (optional) accumulation in a common template of pseudolabeling-based SSL algorithms. We refer to this adapted method (Algorithm 11 in Appendix C.2) as `PabLO`. The high-level steps are also illustrated in Figure 5.1. Next, we discuss the empirical evaluation of `PabLO` and baselines.

5.4 Experiments

We conduct an empirical evaluation over several settings to,

C1. Verify that the adaptations of popular pseudolabeling-based SSL methods with `PabLO` output models with better test accuracy.

C2. Study the effects of choice of error tolerance ϵ on test accuracy of the final model.

C3. Understand the role of pseudolabel accumulation in our method and baselines.

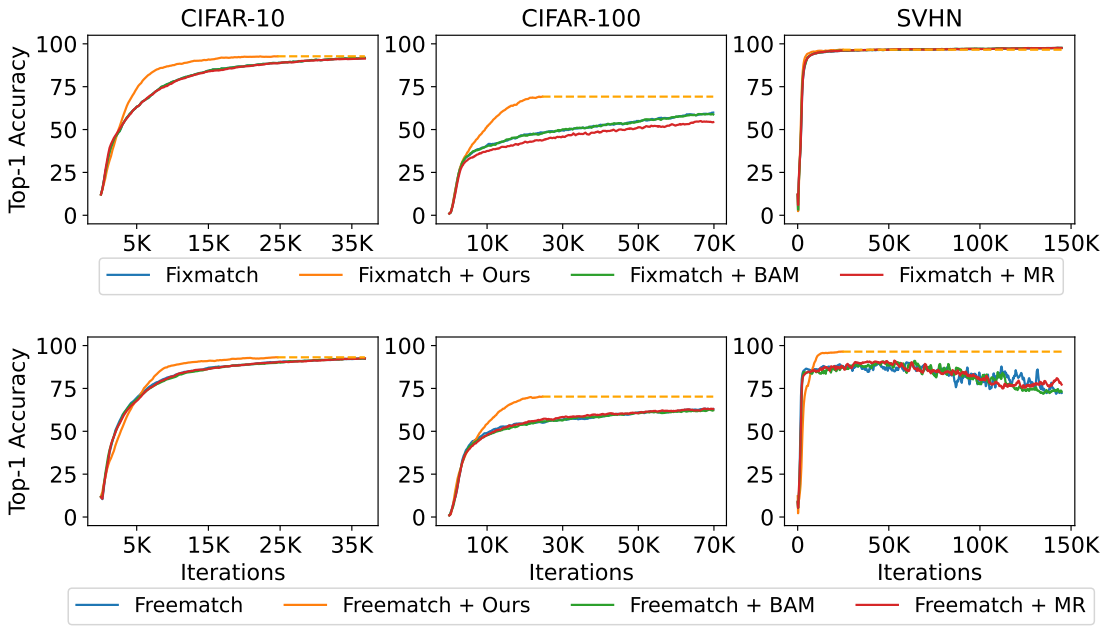


Figure 5.2: Top-1 accuracy of our method and baselines on CIFAR-10, CIFAR-100, and SVHN. We plot the values for every 200 steps.

5.4.1 Experimental Setup

First, we briefly describe the experimental setup, with details deferred to Appendix C.3.

Methods. We use two simple base methods that capture the core ideas of pseudolabeling (PL) and consistency regularization (CR). The first is *Fixmatch* (Sohn et al., 2020), which uses fixed thresholds on (maximum softmax probability) MSP scores for PL and CR. *Freematch* (Wang et al., 2023) improves upon it by using adaptive, class-wise thresholds and class fairness regularization (CFR) along with CR, and is a promising method among others using dynamic thresholds for PL. We include their combinations with recently proposed *Bayesian Model Averaging (BAM)* (Loh et al., 2023) and *Margin Regularization (MR)* [1] (Mishra et al., 2024) to improve calibration in SSL. We replace the pseudolabeling component by our method `PabLO` to obtain *Fixmatch + Ours* (a combination of `PabLO` and CR) and *Freematch + Ours* (a combination of `PabLO`, CR, and CFR). We provide implementations of these in the code submitted along with the paper.

Datasets. We experiment with three datasets: *CIFAR-10* (Krizhevsky et al., 2009b) is an

[1] We assign this name for convenience.

image dataset with 10 classes. *CIFAR-100* (Krizhevsky et al., 2009b) is an extended version of CIFAR-10 with 100 classes. *SVHN* (Netzer et al., 2011) is a 10-class image dataset of digits from Google Street View. More details are summarized in Table 5.1. We use a portion of the validation data (D_{val}) for our method, split into D_{cal}, used to learn the function g, and D_{th}, used to estimate the threshold.

Adjusted iterations for baselines. Empirically, our method requires more time to run compared to base SSL techniques. Therefore, we adopt the following strategy to ensure a fair comparison between the baselines and our method: First, we train our method for 25K iterations and obtain the average per iteration time, denoted as α_o. Then, we train each baseline method b for 5K iterations and obtain the average per iteration time, denoted as α_b. Using these two values, we obtain the adjusted number of iterations, $\frac{\alpha_o}{\alpha_b} \times 25000$, for the baseline method b. Coincidentally, baselines under the same dataset have similar runtimes. We, therefore, set the adjusted number of iterations on a dataset level. For CIFAR-10, the adjusted number of iterations for baselines is 37,000. For CIFAR-100, the adjusted number of iterations for baselines is 70,000. For SVHN, it is 145,000.

Models and training. The backbone encoder is a Wide ResNet-28-2 for all the datasets. We use the default hyperparameters and dataset-specific settings (learning rates, batch size, optimizers, and schedulers) following previous baseline recommendations (Wang et al., 2022a). For confidence functions class $\mathcal{G}$, we use a class of 2-layer neural nets and provide the last two layers' representations from h as input. We train it using SGD. The hyperparameters are deferred to Appendix C.3. Unless otherwise specified, our method uses pseudolabeling error tolerance $\epsilon = 5\%$.

5.4.2 Results and Discussion

To verify our main claims, we compare the baselines, their combinations with our method, and methods that induce calibrated scores in SSL. We run all methods with three random seeds and report (in Table 5.2) the mean and standard deviation of accuracy across three runs.

Dataset	CIFAR-10	CIFAR-100	SVHN
# Labels	40	400	40
Fixmatch	57.16 ± 12.12	30.38 ± 0.68	62.27 ± 5.57
Fixmatch + MR	54.76 ± 5.92	16.31 ± 1.44	74.41 ± 16.61
Fixmatch + BaM	57.72 ± 1.44	31.45 ± 1.29	76.18 ± 18.61
Fixmatch + Ours	**89.72 ± 5.31**	**67.44 ± 1.14**	**96.58 ± 0.14**
Freematch	79.61 ± 7.24	43.88 ± 3.60	86.87 ± 4.15
Freematch + MR	84.19 ± 6.19	44.20 ± 2.93	84.89 ± 7.82
Freematch + BaM	88.33 ± 0.33	44.49 ± 3.05	81.43 ± 14.80
Freematch + Ours	**90.19 ± 5.37**	**65.37 ± 4.05**	**96.55 ± 0.15**
Softmatch	83.60 ± 7.09	40.73 ± 1.46	59.52 ± 12.47
Softmatch + Ours	**89.96 ± 4.74**	**67.84 ± 0.33**	**96.55 ± 0.10**
Adamatch	75.00 ± 1.10	31.61 ± 1.92	80.83 ± 7.09
Adamatch + Ours	**86.62 ± 10.54**	**67.00 ± 1.02**	**96.55 ± 0.09**

Table 5.2: Top-1 Accuracy for CIFAR-10, CIFAR-100 and SVHN averaged across 3 random seeds. The best accuracy is **bolded.**

C1. Test accuracy improvements. Since our method maximizes the pseudolabeling coverage and accuracy, it provides more accurate pseudolabels for model training. Therefore, we expect it to yield a model with better test accuracy than the baselines. We report the test accuracies at the end of 25K iterations in Table 5.2 for our methods. For the baselines, we report the test accuracies at the end of the corresponding adjusted number of iterations (well above 25K). Figure 5.2 illustrates how the top-1 accuracy evolves during the SSL. Similarly, Figure C.1 and C.2 show how batch pseudolabeling accuracy and batch pseudolabeling coverage change.

First, as expected, integrating our method into the base methods improves test accuracy across all settings. For CIFAR-10, using it with Fixmatch provides almost 2% improvement over Fixmatch alone, and using it with Freematch yields 1% improvement over Freematch. Much more significant improvements are observed in the much harder setting of CIFAR-100: a nearly 10% improvement in top-1 accuracy over Fixmatch and around 5% improvement over Freematch. SVHN is an easier setting; here, the improvements are marginal. With Fixmatch, our performance is similar to that of the baselines. But, using `PabLO` with Freematch improves the performance by 3%.

C2. Error tolerance affects performance. In our method, the error tolerance parameter ϵ

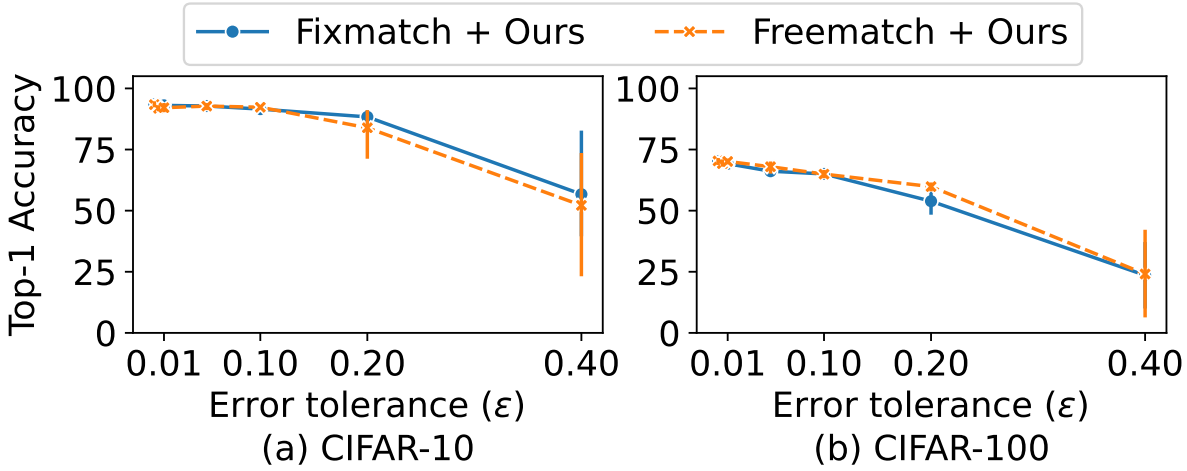


Figure 5.3: Top-1 accuracy of our method with different error tolerance ϵ on (a) CIFAR-10 and (b) CIFAR-100 dataset.

is a knob to control the amount of noise in pseudolabels. A common wisdom in pseudolabeling is that higher noise will lead to worse performance, which is our expectation. To see this, we run our method with $\epsilon \in \{0.01, 0.05, 0.1, 0.2, 0.4\}$ in CIFAR-10 and CIFAR-100 settings, each with three random seeds, and report the results in Figure 5.3. The results are as expected — higher values of ϵ lead to degraded test accuracy due to high noise in the pseudolabels and with decreasing ϵ leads to improved accuracy. These results also suggest that prioritizing the quality (accuracy) of pseudolabels over quantity is a better choice in pseudolabeling. The results are also summarized in Table C.3 and Table C.4 in the Appendix.

To investigate the error tolerance further, we designed error tolerance scheduling using different error tolerances during various stages of SSL training. Table 5.3 summarizes the error tolerance we set at different iterations of SSL training and the corresponding top-1 accuracy for the CIFAR-10 and CIFAR-100 datasets. As we see, starting SSL with a small error tolerance and ending with a large tolerance severely impacts the performance on both CIFAR-10 and CIFAR-100 datasets. While our findings suggest a lower error tolerance is preferable, this may not hold in general. Nevertheless, our framework provides the flexibility to control this explicitly and thus can be tuned by practitioners for the setting at hand.

C3. Is pseudolabel accumulation helpful? Accumulation allows the methods to use old

	CIFAR-10 Top-1 Accuracy	CIFAR-100 Top-1 Accuracy
Schedule 1 Fixmatch + Ours	91.11 ± 1.31	65.78 ± 1.36
Schedule 1 Freematch + Ours	91.09 ± 1.01	66.13 ± 0.48
Schedule 2 Fixmatch + Ours	35.38 ± 29.27	19.56 ± 3.07
Schedule 2 Freematch + Ours	30.48 ± 10.52	24.80 ± 3.49

Table 5.3: Top-1 accuracy for the two error tolerance (ϵ) scheduling. The table reports the ϵ we use between each iteration interval and the top-1 accuracy yielded by the corresponding schedule. For schedule 1, we set $\epsilon = 0.4, 0.2, 0.1, 0.05, 0.01$ when the number of iterations is in the following interval, respectively: $[0, 5\text{K})$, $[5\text{K}, 10\text{K})$, $[10\text{K}, 15\text{K})$, $[15\text{K}, 20\text{K})$, $[20\text{K}, 25\text{K})$. For schedule 2, we set $\epsilon = 0.01, 0.05, 0.1, 0.2, 0.4$, for the same intervals, respectively.

pseudolabels for points that couldn't get pseudolabeled in the current iteration. Thus, we expect accumulation to help improve the utilization of unlabeled data and lead to better test accuracy in cases where the pseudolabel quality is assured to be high in all iterations. We run two variations of our method and baselines — with and without accumulation and report the results in Table 5.4. We observe that our method has similar test accuracy irrespective of accumulation. However, accumulation achieves better coverage in early iterations, as observed in Figure C.3 in Appendix C.3. These results are unsurprising since our method ensures high quality of pseudolabels while maximizing coverage; it can eventually catch up with the version using accumulation, leading to similar final test accuracies. On the other hand, having accumulation hurts the performance of baseline models. This might be because the pseudolabels generated by the baseline models are inaccurate, especially in the earlier iterations, thus degrading the overall performance. Overall, we believe accumulation will be helpful when we have pseudolabels with high accuracy. The plots for pseudolabeling coverage and accuracy over the entire run are in Figures C.3, C.4 in Appendix C.3.

Method	Acc—True	Acc—False
Fixmatch	67.62 ± 2.10	90.08 ± 0.78
Fixmatch + MR	64.78 ± 4.64	90.41 ± 0.83
Fixmatch + BaM	68.10 ± 2.02	90.67 ± 0.90
Freematch	85.40 ± 1.36	92.26 ± 0.18
Freematch + MR	83.59 ± 2.59	92.17 ± 0.36
Freematch + BaM	85.48 ± 3.02	92.32 ± 0.25
Fixmatch + Ours	**92.69 ± 0.74**	**92.80 ± 0.56**
Freematch + Ours	**93.10 ± 0.28**	**91.80 ± 1.08**

Table 5.4: Results on CIFAR-10 with and without pseudolabel accumulation (Acc) for all the methods.

5.5 Related Work

Semi-supervised learning (SSL). There is a rich literature on SSL (Zhu, 2005; Chapelle et al., 2006; Singh et al., 2008; Oliver et al., 2018). This literature comprises of a wide variety of approaches. Among these, significant focus has been placed on self-training (also called pseudolabeling) (Scudder, 1965; Blum and Mitchell, 1998; Rosenberg et al., 2005; Lee et al., 2013; Oymak and Gulcu, 2020; Amini et al., 2023), generative models (Nigam et al., 2000; Adams and Ghahramani, 2009; Kingma et al., 2014), graph-based strategies (Blum and Chawla, 2001; Niyogi, 2013; Subramanya and Talukdar, 2022), and transductive approaches (Vapnik et al., 1998; Joachims, 1999). Due to their simplicity, pseudolabeling-based approaches have gained prominence and are widely used in application areas such as NLP (Karamanolakis et al., 2021), speech recognition (Kahn et al., 2020), and protein prediction (El-Manzalawy et al., 2016). Our paper focuses on recent variants of this, discussed next.

Pseudolabeling based SSL. These methods generate artificial labels for unlabeled data and use them for training the model. A crucial challenge here is the issue of confirmation bias (Arazo et al., 2020), i.e., when a model starts to reinforce its own mistakes. To overcome this and to maintain a high quality of pseudolabels, confidence-based thresholding is applied. Here, only the unlabeled data with confidence higher than a particular threshold is used (Sohn et al., 2020). Due to the limitations of fixed thresholds, adaptive thresholds based on the classifier's learning status have been introduced to improve performance (Xu et al., 2021; Zhang et al., 2021b; Wang

et al., 2023). Nearly all of these methods also use some form of consistency regularization (Laine and Aila, 2017; Bachman et al., 2014; Sajjadi et al., 2016; Fan et al., 2021; Kukačka et al., 2017) where the core idea is that the model should produce similar prediction when presented with different versions (perturbations) of inputs and all the present SSL methods (Xie et al., 2020; Wang et al., 2023; Sohn et al., 2020; Zhang et al., 2021b; Chen et al., 2023b; Xu et al., 2021).

Confidence functions and calibration. Miscalibration (overconfidence) in neural networks plagues various applications (Nguyen et al., 2015a; Hendrycks and Gimpel, 2017b; Guo et al., 2017), including SSL. To mitigate this in general, a range of solutions have been proposed, including training-time methods (Moon et al., 2020; Kumar et al., 2018; Hui et al., 2023; Corbière et al., 2019; Foret et al., 2021) and post-hoc methods (Guo et al., 2017; Kumar et al., 2019; Gupta and Ramdas, 2022; Kull et al., 2019; Zadrozny and Elkan, 2002). In pseudolabeling based SSL, recent works (Rizve et al., 2021; Loh et al., 2023; Mishra et al., 2024) noted the issue of miscalibration. To promote calibration, Loh et al. (2023) use Bayesian neural nets by replacing the model's final layer with a Bayesian layer. Rizve et al. (2021) utilize negative labels and an uncertainty-aware pseudolabel selection technique. Mishra et al. (2024) incorporate a regularizer to encourage calibration.

While calibration is a reasonable goal in general, it may not be sufficient to address the overconfidence problem in SSL and other applications. In pseudolabeling, we seek the use of scores that can easily segregate the model's correct and incorrect predictions, which is closely related to the ordinal ranking criterion (Hendrycks and Gimpel, 2017b; Moon et al., 2020; Foret et al., 2021; Corbière et al., 2019). Rather than experimenting with several such choices, ideally, we would have a flexible framework that can learn confidence functions explicitly optimizing pseudolabeling objectives.

5.6 Conclusion

Common semi-supervised learning (SSL) methods rely on pseudolabeling, but their effectiveness is limited by unreliable confidence scores and heuristic thresholding strategies. We address these

issues by introducing a principled framework for learning confidence scores and thresholds with explicit control over pseudolabeling error. We adapt existing SSL methods with this framework and empirically show that the adapted methods achieve a higher test accuracy compared to their standard versions. Additionally, we introduce pseudolabel accumulation and analyze its impact, showing that it benefits methods with reliable pseudolabels, such as those using our framework. In sum, by providing a principled, data-driven approach to obtaining scores and thresholds for pseudolabeling, our work enhances SSL methods and opens the door to more reliable and efficient pseudolabeling-based SSL.

Chapter 6

Weak Supervision and Structured Prediction

Weak supervision (WS) is a rich set of techniques that produce pseudolabels by aggregating easily obtained but potentially noisy label estimates from a variety of sources. WS is theoretically well understood for binary classification, where simple approaches enable consistent estimation of pseudolabel noise rates. Using this result, it has been shown that downstream models trained on the pseudolabels have generalization guarantees nearly identical to those trained on clean labels. While this is exciting, users often wish to use WS for *structured prediction*, where the output space consists of more than a binary or multi-class label set: e.g., rankings, graphs, manifolds, and more. Do the favorable theoretical properties of WS for binary classification lift to this setting? We answer this question in the affirmative for a wide range of scenarios. For labels taking values in a finite metric space, we introduce techniques new to weak supervision based on pseudo-Euclidean embeddings and tensor decompositions, providing a nearly-consistent noise rate estimator. For labels in constant-curvature Riemannian manifolds, we introduce new invariants that also yield consistent noise rate estimation. In both cases, when using the resulting pseudolabels in concert with a flexible downstream model, we obtain generalization guarantees nearly identical to those for models trained on clean data. Several of our results, which can be viewed as robustness

guarantees in structured prediction with noisy labels, may be of independent interest. Empirical evaluation validates our claims and shows the merits of the proposed method.

6.1 Introduction

Weak supervision (WS) is an array of methods used to construct pseudolabels for training supervised models in label-constrained settings. The standard workflow (Ratner et al., 2016, 2018; Fu et al., 2020) is to assemble a set of cheaply-acquired labeling functions—simple heuristics, small programs, pretrained models, knowledge base lookups—that produce multiple noisy estimates of what the true label is for each unlabeled point in a training set. These noisy outputs are modeled and aggregated into a single higher-quality pseudolabel. Any conventional supervised end model can be trained on these pseudolabels. This pattern has been used to deliver excellent performance in a range of domains in both research and industry settings (Dunnmon et al., 2020; Ré et al., 2020; Safranchik et al., 2020), bypassing the need to invest in large-scale manual labeling. Importantly, these successes are usually found in binary or small-cardinality classification settings.

While exciting, users often wish to use weak supervision in *structured prediction* (SP) settings, where the output space consists of more than a binary or multiclass label set (Bakir et al., 2007; Kuleshov and Liang, 2015). In such cases, there exists a meaningful algebraic or geometric structure to exploit. Structured prediction includes, for example, learning rankings used for recommendation systems (Korba and Alexandre Garcia, 2018), regression in metric spaces (Petersen and Müller, 2019), learning on manifolds (Rudi et al., 2018), graph-based learning (Graber and Schwing, 2019), and more.

An important advantage of WS in the standard setting of binary classification is that it sometimes yields models with nearly the same generalization guarantees as their fully-supervised counterparts. Indeed, the penalty for using pseudolabels instead of clean labels is only a multiplicative constant. This is a highly favorable tradeoff since acquiring more unlabeled data is easy. This property leads us to ask the key question for this work: **does weak supervision for structured prediction preserve generalization guarantees?** We answer this question in the affirmative, justifying

the application of WS to settings far from its current use.

Generalization results in WS rely on two steps (Ratner et al., 2019; Fu et al., 2020): (i) showing that the estimator used to learn the model of the labeling functions is consistent, thus recovering the noise rates for these noisy voters, and (ii) using a noise-aware loss to de-bias end-model training (Natarajan et al., 2013a). Lifting these two results to structured prediction is challenging. The only available weak supervision technique suitable for SP is that of (Shin et al., 2022a). It suffers from several limitations. First, it relies on the availability of isometric embeddings of metric spaces into $\mathbb{R}^d$—but does not explain how to find these. Second, it does not tackle downstream generalization at all. We resolve these two challenges.

We introduce results for a wide variety of structured prediction problems, requiring only that the labels live in some metric space. We consider both finite and continuous (manifold-valued) settings. For finite spaces, we apply two tools that are new to weak supervision. The approach we propose combines isometric *pseudo-Euclidean embeddings* with *tensor decompositions*—resulting in a nearly-consistent noise rate estimator. In the continuous case, we introduce a label model suitable for the so-called *model spaces*—Riemannian manifolds of constant curvature—along with extensions to even more general spaces. In both cases, we show generalization results when using the resulting pseudolabels in concert with a flexible end model from (Ciliberto et al., 2016; Rudi et al., 2018).

Contributions:

- New techniques for performing weak supervision in finite metric spaces based on isometric pseudo-Euclidean embeddings and tensor decomposition algorithms,
- Generalizations to manifold-valued regression in constant-curvature manifolds,
- Finite-sample error bounds for noise rate estimation in each scenario,
- Generalization error guarantees for training downstream models on pseudolabels,
- Experiments confirming the theoretical results and showing improvements over (Shin et al.,

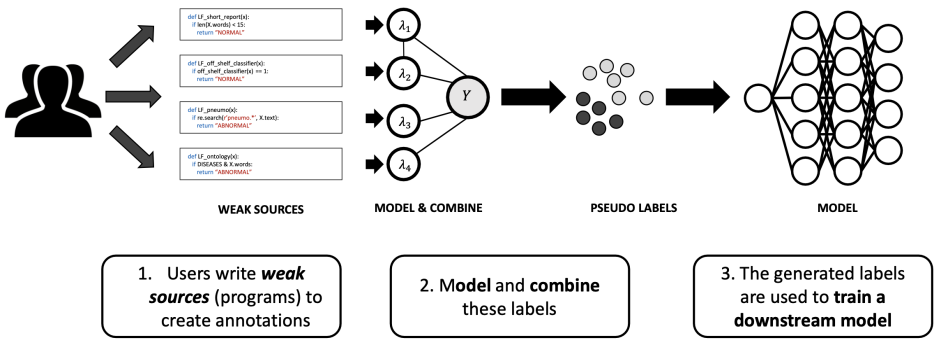


Figure 6.1: Weak supervision pipeline.

2022a).

6.2 Background and Problem Setup

Our goal is to theoretically characterize how well learning with pseudolabels (built with weak supervision techniques) performs in structured prediction. We seek to understand the interplay between the noise in WS sources and the generalization performance of the downstream structured prediction model. We provide a brief background and introduce our problem and some useful notation.

6.2.1 Structured Prediction

Structured prediction (SP) involves predicting labels in spaces with rich structure. Denote the label space by $\mathcal{Y}$. Conventionally $\mathcal{Y}$ is a set, e.g., $\mathcal{Y} = \{-1, +1\}$ for binary classification. In the SP setting, $\mathcal{Y}$ has some additional algebraic or geometric structure. In this work, we assume that $\mathcal{Y}$ is a metric space with metric (distance) $d_{\mathcal{Y}}$. This covers many types of problems, including

- Rankings, where $\mathcal{Y} = S_\rho$, the symmetric group on $\{1, \ldots, \rho\}$, i.e., labels are permutations,
- Graphs, where $\mathcal{Y} = \mathcal{G}_\rho$, the space of graphs with vertex set $V = \{1, \ldots, \rho\}$,
- Riemannian manifolds, including $\mathcal{Y} = \mathbb{S}_d$, the sphere, or $\mathbb{H}_d$, the hyperboloid.

Learning and Generalization in Structured Prediction In conventional supervised learning we have a dataset $\{(x_1, y_1), \ldots, (x_n, y_n)\}$ of i.i.d. samples drawn from distribution ρ over $\mathcal{X} \times \mathcal{Y}$. As usual, we seek to learn a model that generalizes well to points not seen during training. Let $\mathcal{F} = \{f : \mathcal{X} \mapsto \mathcal{Y}\}$ be a family of functions from $\mathcal{X}$ to $\mathcal{Y}$. Define the risk $R(f)$ for $f \in \mathcal{F}$ and f^* as

$$R(f) = \int_{\mathcal{X}\times\mathcal{Y}} d^2_{\mathcal{Y}}(f(x), y) d\rho(x, y) \qquad f^* \in \arg\min_{f\in\mathcal{F}} R(f). \tag{6.1}$$

For a large class of settings (including all of those we consider in this paper), (Ciliberto et al., 2016; Rudi et al., 2018) have shown that the estimator $\hat{f}$ is consistent:

$$\hat{f}(x) = \arg\min_{y\in\mathcal{Y}} F(x, y) \qquad F(x, y) := \frac{1}{n}\sum_{i=1}^{n} \alpha_i(x) d^2_{\mathcal{Y}}(y, y_i), \tag{6.2}$$

where $\alpha(x) = (\mathbf{K} + \nu\mathbf{I})^{-1}\mathbf{K}_x$. Here, $\mathbf{K}$ is the kernel matrix for a p.d. kernel $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$, so that $\mathbf{K}_{i,j} = k(x_i, x_j)$, $(\mathbf{K}_x)_i = k(x, x_i)$, and ν is a regularization parameter. The procedure here is to first compute the weights α and then to perform the optimization in (6.2) to make a prediction.

An exciting contribution of (Ciliberto et al., 2016; Rudi et al., 2018) is the generalization bound

$$R(\hat{f}) \leq R(f^*) + \mathcal{O}(n^{-\frac{1}{4}}),$$

that holds with high probability, as long as there is no label noise. The key question we tackle is *does the use of pseudolabels instead of true labels y_i affect the generalization rate?*

6.2.2 Weak Supervision

In WS, we cannot access *any* of the ground-truth labels y_i. Instead we observe for each x_i the noisy votes $\lambda_{1,i}, \ldots, \lambda_{m,i}$. These are m weak supervision outputs provided by *labeling functions* (LFs) s_a, where $s_a : \mathcal{X} \to \mathcal{Y}$ and $\lambda_{a,i} = s_a(x_i)$. A two-step process is used to construct pseudolabels. First, we learn a *noise model* (also called a label model) that determines how reliable each

source s_a is. That is, we must learn $\boldsymbol{\theta}$ for $P_{\boldsymbol{\theta}}(\lambda_1, \lambda_2, \ldots, \lambda_m | y)$—without having access to any samples of y. Second, the noise model is used to infer a distribution (or its mode) for each point: $P_{\boldsymbol{\theta}}(y_i | \lambda_{1,i}, \ldots, \lambda_{m,i})$.

We adopt the noise model from (Shin et al., 2022a), which is suitable for our SP setting:

$$P_{\boldsymbol{\theta}}(\lambda_1, \ldots, \lambda_m | Y = y) = \frac{1}{Z} \exp\left(-\sum_{a=1}^{m} \theta_a d_Y^2(\lambda_a, y) - \sum_{(a,b)\in E} \theta_{a,b} d_Y^2(\lambda_a, \lambda_b) \right). \tag{6.3}$$

Z is the normalizing partition function, $\boldsymbol{\theta} = [\theta_1, \ldots, \theta_m]^T > 0$ are *canonical* parameters, and E is a set of correlations. The model can be described in terms of the *mean* parameters $\mathbb{E}[d_Y^2(\lambda_a, y)]$. Intuitively, if θ_a is large, the typical distance from λ_a to y is small and the LF is reliable; if θ_a is small, the LF is unreliable. This model is appropriate for several reasons. It is an exponential family model with useful theoretical properties. It subsumes popular special cases of noise, including, for regression, zero-mean multivariate Gaussian noise; for permutations, a generalization of the popular Mallows model; for the binary case, it produces a close relative of the Ising model.

Our goal is to form estimates $\hat{\boldsymbol{\theta}}$ in order to construct pseudolabels. One way to build such pseudolabels is to compute $\tilde{y} = \arg\min_{z \in Y} 1/m \sum_{a=1}^{m} \hat{\theta}_a d_Y^2(z, \lambda_a)$. Observe how the estimated parameters $\hat{\theta}_a$ are used to weight the labeling functions, ensuring that more reliable votes receive a larger weight.

We are now in a position to state the main research question for this work:

Do there exist estimation approaches yielding $\hat{\boldsymbol{\theta}}$ that produce pseudolabels $\tilde{y}$ that maintain the same generalization error rate $\mathcal{O}(n^{-1/4})$ when used in (6.2), or a modified version of (6.2)?

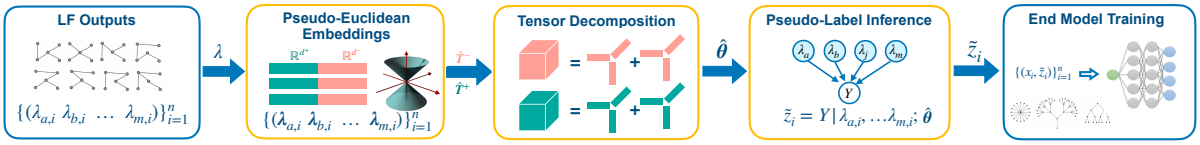


Figure 6.2: Illustration of our weak supervision pipeline for the finite label space setting.

6.3 Noise Rate Recovery in Finite Metric Spaces

In the next two sections, we handle finite metric spaces. Afterwards, we tackle continuous (manifold-valued) spaces. We first discuss learning the noise parameters $\boldsymbol{\theta}$, then the use of pseudolabels.

Roadmap For finite metric spaces with $|\mathcal{Y}| = r$, we apply two tools new to weak supervision. First, we embed $\mathcal{Y}$ into a *pseudo-Euclidean* space (Goldfarb, 1985). These spaces generalize Euclidean space, enabling isometric (distance-preserving) embeddings for any metric. Using pseudo-Euclidean spaces makes our analysis slightly more complex, but we gain the isometry property, which is critical.

Second, we form three-way tensors from embeddings of observed labeling functions. Applying tensor product decomposition algorithms (Anandkumar et al., 2014a), we can recover estimates of the mean parameters $\hat{\mathbb{E}}[d^2_Y(\lambda_a, y)]$ and ultimately $\hat{\theta}_a$. Finally, we reweight the model (6.2) to preserve generalization.

The intuition behind this approach is the following. First, we need a technique that can provide consistent or nearly-consistent estimates of the parameters in the noise model. Second, we need to handle any finite metric space. Techniques like the one introduced in (Fu et al., 2020) handle the first—but do not work for generic finite metric spaces, only binary labels and certain sequences. Techniques like the one in (Shin et al., 2022a) handle any metric space—but only have consistency guarantees in highly restrictive settings (e.g., it requires an isometric embedding, that the distribution over the resulting embeddings is isomorphic to certain distributions, and the true label only takes on two values). Pseudo-Euclidean embeddings used with tensor decomposition algorithms meet both requirements

6.3.1 Pseudo-Euclidean Embeddings

Our first task is to embed the metric space into a continuous space—enabling easier computation and potential dimensionality reduction. A standard approach is multi-dimensional scaling (MDS) (Kruskal and Wish, 1978), which embeds $\mathcal{Y}$ into $\mathbb{R}^d$. A downside of MDS is that not all metric spaces embed (isometrically) into Euclidean space, as the square distance matrix $\mathbf{D}$ must be positive semi-definite.

A simple and elegant way to overcome this difficulty is to instead use *pseudo-Euclidean* spaces for embeddings. These pseudo-spaces do not require a p.s.d. inner product. As an outcome, any finite metric space can be embedded into a pseudo-Euclidean space with *no distortion* (Goldfarb, 1985)—so that distances are exactly preserved. Such spaces have been applied to similarity-based learning methods (Pękalska et al., 2001; Laub et al., 2006; Pękalska et al., 2006). A vector $\mathbf{u}$ in a pseudo-Euclidean space $\mathbb{R}^{d^+,d^-}$ has two parts: $\mathbf{u}^+ \in \mathbb{R}^{d^+}$ and $\mathbf{u}^- \in \mathbb{R}^{d^-}$. The dot product and the squared distance between any two vectors $\mathbf{u}, \mathbf{v}$ are $\langle \mathbf{u}, \mathbf{v} \rangle_\phi = \langle \mathbf{u}^+, \mathbf{v}^+ \rangle - \langle \mathbf{u}^-, \mathbf{v}^- \rangle$ and $d^2_\phi(\mathbf{u}, \mathbf{v}) = ||\mathbf{u}^+ - \mathbf{v}^+||_2^2 - ||\mathbf{u}^- - \mathbf{v}^-||_2^2$. These properties enable isometric embeddings: the distance can be decomposed into two components that are individually induced from p.s.d. inner products—and can thus be embedded via MDS. Indeed, pseudo-Euclidean embeddings effectively run MDS for each component (see Algorithm 3 steps 4-9). To recover the original distance, we obtain $||\mathbf{u}^+ - \mathbf{v}^+||_2^2$ and $||\mathbf{u}^- - \mathbf{v}^-||_2^2$ and subtract.

Example: To see why such embeddings are advantageous, we compare with a one-hot vector representation (whose dimension is $|\mathcal{Y}|$). Consider a tree with a root node and three branches, each of which is a path with t nodes. Let $\mathcal{Y}$ be the nodes in the tree with the shortest-hops distance as the metric. The pseudo-Euclidean embedding dimension is just $d = 3$; see the Appendix for more details. The one-hot embedding dimension is $d = |\mathcal{Y}| = 3t + 1$—arbitrarily larger!

Now we are ready to apply these embeddings to our problem. Abusing notation, we write $\boldsymbol{\lambda}_a$ and $\mathbf{y}$ for the pseudo-Euclidean embeddings of λ_a, y, respectively. We have that $d^2_{\mathcal{Y}}(\lambda_a, y) = d^2_\phi(\boldsymbol{\lambda}_a, \mathbf{y})$, so that there is no loss of information from working with these spaces. In addition, we write the mean as $\boldsymbol{\mu}_{a,y} = \mathbb{E}[\boldsymbol{\lambda}_a | \mathbf{y}]$ and the covariance as $\boldsymbol{\Sigma}_{a,y}$. Our goal is to obtain an accurate estimate

$\hat{\boldsymbol{\mu}}_{a,y} = \hat{\mathbb{E}}[\boldsymbol{\lambda}_a|\mathbf{y}]$, which we will use to estimate the mean parameters $\mathbb{E}[d_Y^2(\lambda_a, y)]$. If we could observe y, it would be easy to empirically estimate $\boldsymbol{\mu}_{a,y}$—but we do not have access to it. Our approach will be to apply tensor decomposition for multi-view mixtures (Anandkumar et al., 2014b).

6.3.2 Multi-View Mixtures and Tensor Decompositions

In a multi-view mixture model, multiple views $\{\lambda_a\}_{a=1}^m$ of a latent variable Y are observed. These views are independent when conditioned on Y. We treat the positive and negative components $\boldsymbol{\lambda}_a^+ \in \mathbb{R}^{d^+}$ and $\boldsymbol{\lambda}_a^- \in \mathbb{R}^{d^-}$ of our pseudo-Euclidean embedding as separate multi-view mixtures:

$$\boldsymbol{\lambda}_a^+|\mathbf{y} \sim \boldsymbol{\mu}_{a,y}^+ + \sigma\sqrt{d^+} \cdot \boldsymbol{\epsilon}_a^+ \quad \text{and} \quad \boldsymbol{\lambda}_a^-|\mathbf{y} \sim \boldsymbol{\mu}_{a,y}^- + \sigma\sqrt{d^-} \cdot \boldsymbol{\epsilon}_a^- \qquad \forall a \in [m], \tag{6.4}$$

where $\boldsymbol{\mu}_{a,y}^+ = \mathbb{E}[\boldsymbol{\lambda}_a^+|\mathbf{y}]$, $\boldsymbol{\mu}_{a,y}^- = \mathbb{E}[\boldsymbol{\lambda}_a^-|\mathbf{y}]$ and $\boldsymbol{\epsilon}_a^+, \boldsymbol{\epsilon}_a^-$ are mean zero random vectors with covariances $\frac{1}{d^+}\mathbf{I}_{d^+}, \frac{1}{d^-}\mathbf{I}_{d^-}$ respectively. Here σ^2 is a proxy variance whose use is described in Assumption 3.

We cannot directly estimate these parameters from observations of $\boldsymbol{\lambda}_a$, due to the fact that $\mathbf{y}$ is not observed. However, we can observe various moments of the outputs of the LFs, such as tensors of outer products of LF triplets. We require that for each a such a triplet exists. Then,

$$\mathbf{T}^+ := \mathbb{E}[\boldsymbol{\lambda}_a^+ \otimes \boldsymbol{\lambda}_b^+ \otimes \boldsymbol{\lambda}_c^+] = \sum_{y \in \mathcal{Y}_s} w_y \boldsymbol{\mu}_{a,y}^+ \otimes \boldsymbol{\mu}_{b,y}^+ \otimes \boldsymbol{\mu}_{c,y}^+ \text{ and } \hat{\mathbf{T}}^+ := \frac{1}{n}\sum_{i=1}^n \boldsymbol{\lambda}_{a,i}^+ \otimes \boldsymbol{\lambda}_{b,i}^+ \otimes \boldsymbol{\lambda}_{c,i}^+. \tag{6.5}$$

Here w_y are the mixture probabilities (prior probabilities of Y) and $\mathcal{Y}_s = \{y : w_y > 0\}$. We similarly define $\mathbf{T}^-$ and $\hat{\mathbf{T}}^-$. We then obtain estimates $\hat{\boldsymbol{\mu}}_{a,y}^+, \hat{\boldsymbol{\mu}}_{a,y}^-$ using an algorithm from (Anandkumar et al., 2014a) with minor modifications to handle pseudo-Euclidean rather than Euclidean space. The overall approach is shown in Algorithm 3. We have three key assumptions for our analysis,

Assumption 1. The support of P_Y, i.e., $k = |\{y : w_y > 0\}|$ and the label space $\mathcal{Y}$ is such that $\min(d^+, d^-) \geq k$, $||\boldsymbol{\mu}_{a,y}^+||_2 = 1, ||\boldsymbol{\mu}_{a,y}^-||_2 = 1$ for $a \in [m], y \in \mathcal{Y}$.

Assumption 2. (Bounded angle between $\boldsymbol{\mu}$ and $\mathbf{y}$) Let $\omega(\mathbf{u}, \mathbf{v})$ denote the angle between any two vectors $\mathbf{u}, \mathbf{v}$ in a Euclidean space. We assume that $\omega(\boldsymbol{\mu}_{a,y}^+, \mathbf{y}^+) \in [0, \pi/2 - c)$, $\omega(\boldsymbol{\mu}_{a,y}^-, \mathbf{y}^-) \in$

Algorithm 3: Algorithm for Pseudolabel Construction

Input: Labeling function outputs $\mathbf{L} = \{(\lambda_{1,i}, \ldots, \lambda_{m,i})\}_{i=1}^{n}$, Label Space $\mathcal{Y} = \{y_0, \ldots, y_{r-1}\}$
Output: Pseudolabels for each data point $\mathbf{Z} = \{\tilde{z}_i\}_{i=1}^{n}$

▷ Step 1: Compute pseudo-Euclidean Embeddings
Construct matrices $\mathbf{D} \in \mathbb{R}^{r\times r}$, $\mathbf{D}_{ij} = d_{\mathcal{Y}}^2(y_i, y_j)$ and $\mathbf{M} \in \mathbb{R}^{r\times r}$, $\mathbf{M}_{ij} = \frac{1}{2}(\mathbf{D}_{0i}^2 + \mathbf{D}_{0j}^2 - \mathbf{D}_{ij}^2)$
Compute eigendecomposition of $\mathbf{M}$ and let $\mathbf{M} = \mathbf{U}\mathbf{C}\mathbf{U}^T$
Set l^+, l^- be indices of positive and negative eigenvalues sorted by their magnitude
Let $d^+ = |l^+|, \quad d^- = |l^-|$ i.e. the sizes of lists l^+ and l^- respectively.
Construct permutation matrix $\mathbf{I}_{perm} \in \mathbb{R}^{r\times(d^+ + d^-)}$ by concatenating l^+, l^- in order
$\bar{\mathbf{C}} = \mathbf{C}\mathbf{I}_{perm}, \bar{\mathbf{U}} = \mathbf{U}\mathbf{I}_{perm}$
$\mathbb{Y} = \bar{\mathbf{U}}^T \bar{\mathbf{C}}^{\frac{1}{2}} \in \mathbb{R}^{r\times(d^+ + d^-)}$ and let this define the mapping $g : \mathcal{Y} \mapsto \mathbb{Y}$

▷ Step 2: Parameter Estimation Using Tensor Decomposition
for $a \leftarrow 1$ to $m - 3$ **do**
    Obtain embeddings $\boldsymbol{\lambda}_{a,i} = g(\lambda_{a,i}), \boldsymbol{\lambda}_{b,i} = g(\lambda_{b,i}), \boldsymbol{\lambda}_{c,i} = g(\lambda_{c,i}) \quad \forall i \in [n]$ where a, b, c are uncorrelated
    Construct tensors $\hat{\mathbf{T}}^+$ and $\hat{\mathbf{T}}^-$ as defined in (6.5) for triplet (a, b, c)
    $\hat{\boldsymbol{\mu}}_{a,y}^+, \hat{\boldsymbol{\mu}}_{b,y}^+, \hat{\boldsymbol{\mu}}_{c,y}^+ = \texttt{TensorDecomposition}(\hat{\mathbf{T}}^+)$
    $\hat{\boldsymbol{\mu}}_{a,y}^-, \hat{\boldsymbol{\mu}}_{b,y}^-, \hat{\boldsymbol{\mu}}_{c,y}^- = \texttt{TensorDecomposition}(\hat{\mathbf{T}}^-)$
    $s_{a,y}^+ = \min_{z\in\{-1,+1\}} \phi(z \cdot \hat{\boldsymbol{\mu}}_{a,y}^+, \mathbf{y}^+)$ and similarly $s_{b,y}^+, s_{c,y}^+, s_{a,y}^-, s_{b,y}^-, s_{c,y}^-$
    $\hat{\boldsymbol{\mu}}_{a,y}^+ = s_{a,y}^+ \cdot \hat{\boldsymbol{\mu}}_{a,y}^+$ and similarly correct signs of $\hat{\boldsymbol{\mu}}_{b,y}^+, \hat{\boldsymbol{\mu}}_{c,y}^+, \hat{\boldsymbol{\mu}}_{a,y}^-, \hat{\boldsymbol{\mu}}_{b,y}^-, \hat{\boldsymbol{\mu}}_{c,y}^-$
end for

▷ Step 3: Infer Pseudo-Labels
$\tilde{Z}^{(i)} = \tilde{z}_i \sim Y | \lambda_a = \lambda_a^{(i)}, \ldots \lambda_m = \lambda_m^{(i)}; \hat{\boldsymbol{\theta}}$

return $\{\tilde{z}_i\}_{i=1}^{n}$

$[0, \pi/2 - c)$ $\forall a \in [m]$, and $y \in \mathcal{Y}_s$, for some sufficiently small $c \in (0, \pi/4]$ such that $\sin(c) \geq \max(\epsilon_0(d^+), \epsilon_0(d^-))$, where $\epsilon_0(d)$ is defined for some $n > n_0$ samples in (6.6).

Assumption 3. σ is such that the recovery error with model (6.4) is at least as large as with (6.3) .

These enable providing guarantees on recovering the mean vector magnitudes (1) and signs (2) and simplify the analysis (1), (3); all three can be relaxed at the expense of a more complex analysis.

Our first theoretical result shows that we have near-consistency in estimating the mean parameters in (6.3). We use standard notation $\tilde{\mathcal{O}}$, ignoring logarithmic factors.

Theorem 6.3.1. *Let $\hat{\boldsymbol{\mu}}^+_{a,y}, \hat{\boldsymbol{\mu}}^-_{a,y}$ be the estimates of $\boldsymbol{\mu}^+_{a,y}, \boldsymbol{\mu}^-_{a,y}$ returned by Algorithm 3 with input $\hat{\mathbf{T}}^+, \hat{\mathbf{T}}^-$ constructed using isometric pseudo-Euclidean embeddings (in $\mathbb{R}^{d^+,d^-}$). Suppose Assumptions 1 and 2 are met, a sufficiently large number of samples n are drawn from the model in (6.3), and $k = |\mathcal{Y}_s|$. Then there exists a constant $C_0 > 0$ such that with high probability $\forall a \in [m]$ and $y \in \mathcal{Y}_s$,*

$$|\theta_a - \hat{\theta}_a| \leq C_0 \left| \mathbb{E}[d^2_{\mathcal{Y}}(\lambda_a, y)] - \hat{\mathbb{E}}[d^2_{\mathcal{Y}}(\lambda_a, y)] \right| \leq \epsilon(d^+) + \epsilon(d^-),$$

where

$$\epsilon(d) := \begin{cases} \tilde{\mathcal{O}}\left(k\sqrt{\frac{d}{n}}\right) + \tilde{\mathcal{O}}\left(\frac{\sqrt{k}}{d}\right) & \text{if } \sigma^2 = \Theta(1), \\ \tilde{\mathcal{O}}\left(\sqrt{\frac{k}{n}}\right) + \tilde{\mathcal{O}}\left(\frac{\sqrt{k}}{d}\right) & \text{if } \sigma^2 = \Theta(\frac{1}{d}). \end{cases} \tag{6.6}$$

We interpret Theorem 6.3.1. It is a nearly direct application of (Anandkumar et al., 2014b). There are two noise cases for σ. In the high-noise case, σ is independent of dimension d (and thus $|\mathcal{Y}|$). Intuitively, this means the average distance balls around each LF begin to overlap as the number of points grows—explaining the multiplicative k term. If the noise scales down as we add more embedded points, this problem is removed, as in the low-noise case. In both cases, the second error term comes from using the algorithm of (Anandkumar et al., 2014a) and is independent of the sampling error. Since $k = \Theta(d)$, this term goes down with d. The first error term is due to sampling noise and goes to zero in the number of samples n. Note the tradeoffs of using the embeddings. If we used one-hot encoding, $d = |\mathcal{Y}|$, and in the high-noise case, we would pay a very heavy cost for $\sqrt{d/n}$. However, while sampling error is minimized when using a very small d, we pay a cost in the second error term. This leads to a tradeoff in selecting the appropriate embedding dimension.

6.4 Generalization Error for Structured Prediction in Finite Metric Spaces

We have access to labeling function outputs $\lambda_{1,i}, \ldots, \lambda_{m,i}$ for points x_i and noise rate estimates $\hat{\theta}_a, \ldots, \hat{\theta}_m$. How can we use these to infer unobserved labels y in (6.2)? Our approach is based on (Natarajan et al., 2013a; van Rooyen and Williamson, 2018), where the underlying loss function is modified to deal with noise. Analogously, we modify (6.2) in such a way that the generalization guarantee is nearly preserved.

6.4.1 Prediction with Pseudolabels

First, we construct the posterior distribution $P_{\hat{\boldsymbol{\theta}}}(Y = y|\lambda)$. We use our estimated noise model $P_{\hat{\boldsymbol{\theta}}}(\lambda|Y)$ and the prior $P(Y = y)$. We create pseudo-labels for each data point by drawing a random sample from the posterior distribution conditioned on the output of labeling functions: $\tilde{Z}^{(i)} = \tilde{z}_i \sim Y|\lambda_a = \lambda_a^{(i)}, \ldots, \lambda_m = \lambda_m^{(i)}; \hat{\boldsymbol{\theta}}$. We thus observe $(x_1, \tilde{z}_1), \ldots, (x_n, \tilde{z}_n)$ where $\tilde{z}_i$ is sampled as above. To overcome the effect of noise, we create a perturbed version of the distance function using the noise rates, generalizing (Natarajan et al., 2013a). This requires us to characterize the noise distribution induced by our inference procedure. In particular, we seek the probability that $\tilde{Z} = y_j$ when the true label is y_j. This can be expressed as follows. Let $\mathcal{Y}^m$ denote the m-fold Cartesian product of $\mathcal{Y}$ and let $\Lambda_u = (\lambda_1^{(u)}, \ldots, \lambda_m^{(u)})$ denote its u^{th} entry. We write

$$\mathbf{P}_{ij} = P_{\boldsymbol{\theta}}(\tilde{Z} = y_j|Y = y_i) = \sum_{u=1}^{|\mathcal{Y}^m|} P_{\boldsymbol{\theta}}(\tilde{Z} = y_j|\Lambda = \Lambda^{(u)}) \cdot P_{\boldsymbol{\theta}}(\Lambda = \Lambda^{(u)}|Y = y_i). \tag{6.7}$$

We define $\mathbf{Q}_{ij} = P_{\hat{\boldsymbol{\theta}}}(\tilde{Z} = y_j|Y = y_i)$ using $\hat{\boldsymbol{\theta}}$. $\mathbf{P}$ is the noise distribution induced by the true parameters $\boldsymbol{\theta}$ and $\mathbf{Q}$ is an approximation obtained from inference with the *estimated* parameters $\hat{\boldsymbol{\theta}}$. With this terminology, we can define the perturbed version of the distance function and a corresponding replacement of (6.2):

$$\tilde{d}_q(T, \tilde{Y} = y_j) := \sum_{i=1}^{k} (\mathbf{Q}^{-1})_{ji} d_Y^2(T, Y = y_i) \quad \forall y_j \in \mathcal{Y}_s, \tag{6.8}$$

$$\tilde{F}_q(x,y) := \frac{1}{n}\sum_{i=1}^{n} \alpha_i(x)\tilde{d}_q(y,\tilde{z}_i) \qquad \hat{f}_q(x) = \arg\min_{y\in\mathcal{Y}} \tilde{F}_q(x,y). \tag{6.9}$$

We similarly define $\tilde{d}_p, \tilde{F}_p, \hat{f}_p$ using the true noise distribution $\mathbf{P}$. The perturbed distance $\tilde{d}_p$ is an unbiased estimator of the true distance. However, we do not know the true noise distribution $\mathbf{P}$ hence we cannot use it for prediction. Instead we use $\tilde{d}_q$. Note that $\tilde{d}_q$ is no longer an unbiased estimator—its bias can be expressed as a function of the parameter recovery error bound in Theorem 6.3.1.

6.4.2 Bounding the Generalization Error

What can we say about the excess risk $R(\hat{f}_q) - R(f^*)$? Note that compared to the prediction based on clean labels, there are two additional sources of error. One is the noise in the labels (i.e., even if we know the true $\mathbf{P}$, the quality of the pseudolabels is imperfect). The other is our estimation procedure for the noise distribution. We must address both sources of error.

Our analysis uses the following assumptions on the minimum and maximum singular values $\sigma_{\min}(\mathbf{P})$, $\sigma_{\max}(\mathbf{P})$ and the condition number $\kappa(\mathbf{P})$ of true noise matrix $\mathbf{P}$ and the function F. Additional detail is provided in the Appendix.

Assumption 4. (Noise model is not arbitrary) The true parameters $\boldsymbol{\theta}$ are such that $\sigma_{\min}(\mathbf{P}) > 0$, and the condition number $\kappa(\mathbf{P})$ is sufficiently small.

Assumption 5. (Normalized features) $|\alpha(x)| \leq 1$, for all $x \in \mathcal{X}$.

Assumption 6. (Proxy strong convexity) The function F in (6.2) satisfies the following property with some $\beta > 0$. As we move away from the minimizer of F, the function increases, and the rate of increase is proportional to the distance between the points:

$$F\big(x, f(x)\big) \geq F\big(x, \hat{f}(x)\big) + \beta \cdot d_{\mathcal{Y}}^2\big(f(x), \hat{f}(x)\big) \qquad \forall x \in \mathcal{X}, \forall f \in \mathcal{F}. \tag{6.10}$$

With these assumptions, we provide a generalization result for prediction with pseudolabels,

Theorem 6.4.1. *(Generalization Error) Let $\hat{f}$ be the minimizer as defined in (6.2) over the clean labels and let $\hat{f}_q$ (defined in (6.9)) be the minimizer over the noisy labels obtained from inference in Algorithm 3. Suppose Assumptions 4,5,6 hold. Then for $\epsilon_2 = k^{5/2} \cdot \tilde{\mathcal{O}}(\epsilon(d^+) + \epsilon(d^-)) \cdot \left(1 + \frac{\kappa(\mathbf{P})}{\sigma_{\min}(\mathbf{P})}\right)$ and $c_1 = 1 + \frac{\sqrt{k}}{\sigma_{\min}(\mathbf{P})}$, with high probability,*

$$R(\hat{f}_q) \leq R(f^*) + \mathcal{O}(n^{-\frac{1}{4}}) + \tilde{\mathcal{O}}\Big(\frac{c_1}{\beta} n^{-\frac{1}{2}}\Big) + \tilde{\mathcal{O}}\Big(\frac{3\epsilon_2}{\beta} n^{-\frac{1}{2}}\Big). \tag{6.11}$$

Implications and Tradeoffs: We interpret each term in the bound. The first term is present even with access to the clean labels and is hence unavoidable. The second term is the additional error we incur if we learn with the knowledge of the true noise distribution. The third term is due to the use of the estimated noise model. It is dominated by the noise rate recovery result in Theorem 6.3.1. If the third term goes to 0 (perfect recovery) then we obtain the rate $\mathcal{O}(n^{-1/4})$, the same as in the case of access to clean labels. The third term is introduced by our noise rate recovery algorithm and has two terms: one dominated by $\tilde{\mathcal{O}}(n^{-1/2})$ and the other on $\tilde{\mathcal{O}}(\sqrt{k}/d)$ (see discussion of Theorem 6.3.1). Thus, we only pay an extra additive factor $\mathcal{O}(\sqrt{k}/d)$ in the excess risk when using pseudolabels.

6.5 Manifold-Valued Label Spaces: Noise Recovery and Generalization

We introduce a simple recovery method for weak supervision in constant-curvature Riemannian manifolds. First, we briefly introduce some background notation on these spaces, then provide our estimator and consistency result, then the downstream generalization result. Finally, we discuss extensions to symmetric Riemannian manifolds, an even more general class of spaces.

Background on Riemannian manifolds The following is necessarily a very abridged background; more detail can be found in (Lee, 2000; Tu, 2011). A smooth manifold M is a space

where each point is located in a neighborhood diffeomorphic to $\mathbb{R}^d$. Attached to each point $p \in \mathcal{M}$ is a *tangent space* T_pM; each such tangent space is a d-dimensional vector space enabling the use of calculus.

A Riemannian manifold equips a smooth manifold with a Riemannian metric: a smoothly-varying inner product $\langle \cdot, \cdot \rangle_p$ at each point p. This tool allows us to compute angles, lengths, and ultimately, distances $d_{\mathcal{M}}(p, q)$ between points on the manifold as shortest-path distances. These shortest paths are called geodesics and can be parameterized as curves $\gamma(t)$, where $\gamma(0) = p$, or by tangent vectors $v \in T_pM$. The exponential map operation $\exp : T_p\mathcal{M} \to \mathcal{M}$ takes tangent vectors to manifold points. It enables switching between these tangent vectors: $\exp_p(v) = q$ implies that $d_{\mathcal{M}}(p, q) = \|v\|$.

Invariant Our first contribution is a simple invariant that enables us to recover the error parameters. Note that we cannot rely on the finite metric-space technique, since the manifolds we consider have an infinite number of points. Nor do we need an embedding—we have a continuous representation as-is. Instead, we propose a simple idea based on the law of cosines. Essentially, on average, the geodesic triangle formed by the latent variable $y \in \mathcal{M}$ and two observed LFs λ_a, λ_b, is a right triangle. This means it can be characterized by the (Riemannian) version of the Pythagorean theorem:

Lemma 6.5.1. *For $\mathcal{Y} = \mathcal{M}$, a hyperbolic manifold, $y \sim P$ for some distribution P on $\mathcal{M}$ and labeling functions λ_a, λ_b drawn from (6.3), $\mathbb{E} \cosh d_{\mathcal{Y}}(\lambda_a, \lambda_b) = \mathbb{E} \cosh d_{\mathcal{Y}}(\lambda_b, y) \mathbb{E} \cosh d_{\mathcal{Y}}(\lambda_b, y)$, while for $\mathcal{Y} = \mathcal{M}$ a spherical manifold, $\mathbb{E} \cos d_{\mathcal{Y}}(\lambda_a, \lambda_b) = \mathbb{E} \cos d_{\mathcal{Y}}(\lambda_b, y) \mathbb{E} \cos d_{\mathcal{Y}}(\lambda_b, y)$.*

These invariants enable us to easily learn by forming a triplet system. Suppose we construct the equation in Lemma 6.5.1 for three pairs of labeling functions. The resulting system can be solved to express $\mathbb{E}[\cosh(d_{\mathcal{Y}}(\lambda_a, y))]$ in terms of $\mathbb{E} \cosh(d_{\mathcal{Y}}(\lambda_a, \lambda_b)), \mathbb{E} \cosh(d_{\mathcal{Y}}(\lambda_a, \lambda_c)), \mathbb{E} \cosh(d_{\mathcal{Y}}(\lambda_b, \lambda_c))$. Specifically,

$$\mathbb{E} \cosh(d_{\mathcal{Y}}(\lambda_a, y)) = \sqrt{\frac{\mathbb{E} \cosh d_{\mathcal{Y}}(\lambda_a, \lambda_b) \mathbb{E} \cosh d_{\mathcal{Y}}(\lambda_a, \lambda_c)}{(\mathbb{E} \cosh(d_{\mathcal{Y}}(\lambda_b, \lambda_c))^2}}.$$

Note that we can estimate $\hat{\mathbb{E}}$ via the empirical versions of terms on the right, as these are based

on observable quantities. This is a generalization of the binary case in (Fu et al., 2020) and the Gaussian (Euclidean) case in (Shin et al., 2022a) to hyperbolic manifolds. A similar estimator can be obtained for spherical manifolds by replacing cosh with cos.

Using this tool, we can obtain a consistent estimator for θ_a for each of $a = 1, \ldots, m$. Let C_0 satisfy the following inequality $\mathbb{E}|\hat{\mathbb{E}}\cosh(d_{\mathcal{Y}}(\lambda_a, \lambda_b)) - \mathbb{E}\cosh(d_{\mathcal{Y}}(\lambda_a, \lambda_b))| \geq C_0\mathbb{E}|\hat{\mathbb{E}}d^2_{\mathcal{Y}}(\lambda_a, \lambda_b)) - \mathbb{E}d^2_{\mathcal{Y}}(\lambda_a, \lambda_b)|$; that is, C_0 reflects the preservation of concentration when moving from distribution $\cosh(d)$ to d^2. Then,

Theorem 6.5.2. *Let $\mathcal{M}$ be a hyperbolic manifold and $\Delta(\delta) = \min_\rho Pr\Big(\forall i, d_{\mathcal{Y}}(\lambda_{a,i}, \lambda_{b,i)}) \leq \rho\Big) \geq 1 - \delta$ for fixed δ in$(0, 1)$ and . Then, there exists a constant C_1 so that with probability at least $1 - \delta$, $\mathbb{E}|\hat{\mathbb{E}}d^2_{\mathcal{Y}}(\lambda_a, y)) - \mathbb{E}d^2_{\mathcal{Y}}(\lambda_a, y)| \leq C_1\cosh(\Delta(\delta))^{3/2}/C_0\sqrt{2n}$.*

As we hoped, our estimator is consistent. Note that we pay a price for a tighter bound: $\Delta(\delta)$ is large for smaller probability δ. It is possible to estimate the size of $\Delta(\delta)$ (more generally, it is a function of the curvature). We provide more details in the Appendix.

Next, we adapt the downstream model predictor (6.2) in the following way. Let $\hat{\mu}^2_a = \hat{\mathbb{E}}[d^2_{\mathcal{Y}}(\lambda_a, y)]$. Let $\beta = [\beta_1, \ldots, \beta_m]^T$ be such that $\sum_a \beta_a = 1$ and β minimizes $\sum_a \beta^2_a \hat{\mu}^2_a$. Then, we set

$$\tilde{f}(x) = \arg\min_{y \in \mathcal{Y}} \frac{1}{n}\sum_{i=1}^{n} \alpha_i(x) \sum_{a=1}^{m} \beta^2_a d^2_{\mathcal{Y}}(y, \lambda_{a,i}).$$

We simply replace each of the true labels with a combination of the labeling functions. With this, we can state our final result. First, we introduce our assumptions.

Let $q = \arg\min_{z \in \mathcal{Y}} \mathbb{E}[\alpha(x)(y)d^2_{\mathcal{Y}}(z, y)]$, where the expectation is taken over the population level distribution and $\alpha(x)(y)$ denotes the kernel at y.

Assumption 7. (Bounded Hugging Function c.f. (Stromme, 2020)) Let q be defined as above. For all $a, b \in \mathcal{M}$, the hugging function at q is given by $k^b_q(a) = 1 - (\|\log_q(a) - \log_q(b)\|^2 - d^2_{\mathcal{Y}}(a, b))/d^2_{\mathcal{Y}}(q, b)$. We assume that $k^b_q(a)$ is lower bounded by $k_{\min}$.

Assumption 8. (Kernel Symmetry) We assume that for all x and all $v \in T_q\mathcal{M}$, $\alpha(x)(\exp_q(v)) = \alpha(x)(\exp_q(-v))$.

The first condition provides control on how geodesic triangles behave; it relates to the curvature. We provide more details on this in the Appendix. The second assumption restricts us to kernels symmetric about the minimizers of the objective F. Finally, suppose we draw (x, y) and (x', y') independently from P_{XY}. Set $\sigma_o^2 = \alpha(x)(y)\mathbb{E}d_{\mathcal{Y}}^2(y, y')$.

Theorem 6.5.3. *Let $\mathcal{M}$ be a complete manifold and suppose the assumptions above hold. Then, there exist constants C_3, C_4*

$$\mathbb{E}[d_{\mathcal{Y}}^2(\hat{f}(x), \tilde{f}(x))] \leq \frac{C_3\sigma_o^2}{nk_{\min}} + \frac{C_4\sum_{a=1}^m \beta_a^2\hat{\mu}_a^2}{mnk_{\min}}.$$

Note that as both m and n grow, as long as our worst-quality LF has bounded variance, our estimator of the true predictor is consistent. Moreover, we also have favorable dependence on the noise rate. This is because the only error we incur is in computing sub-optimal β coefficients. We comment on this suboptimality in the Appendix.

A simple corollary of Theorem D.4.3 provides the generalization guarantees we sought,

Corollary 6.5.3.1. *Let $\mathcal{M}$ be a complete manifold and suppose the assumptions above hold. Then, with high probability, $R(\tilde{f}) \leq R(f^*) + \mathcal{O}(n^{-\frac{1}{4}})$.*

Extensions to Other Manifolds First, we note that all of our approaches almost immediately lift to products of constant-curvature spaces. For example, we have that $\mathcal{M}_1 \times \mathcal{M}_2$ has metric $d_{\mathcal{Y}}^2(p, q) = d_{\mathcal{M}_1}^2(p_1, q_1) + d_{\mathcal{M}_2}^2(p_2, q_2)$, where p_i, q_i are the projections of p, q onto the ith component.

We can go beyond products of constant-curvature spaces as well. To do so, we can build generalizations of the law of cosines (as needed for the invariance in Lemma 6.5.1). For example, it is possible to do so for symmetric Riemannian manifolds using the tools in (Aslaksen and Huynh, 1991).

6.6 Experiments

Finally, we validate our theoretical claims with experimental results demonstrating improved parameter recovery and end model generalization using our techniques over that of prior work

(Shin et al., 2022a). We illustrate both the finite metric space and continuous space cases by targeting rankings (i.e., permutations) and hyperbolic spaces. In the case of rankings, we show that our pseudo-Euclidean embeddings with tensor decomposition estimator yield stronger parameter recovery and downstream generalization than Shin et al. (2022a). In the case of hyperbolic regression (an example of a Riemannian manifold), we show that our estimator yields improved parameter recovery over Shin et al. (2022a).

Finite metric spaces: Learning to rank To experimentally evaluate our tensor decomposition estimator for finite metric spaces, we consider the problem of learning to rank. We construct a synthetic dataset whose ground truth comprises n samples of two distinct rankings among the finite metric space of all length-four permutations. We construct three labeling functions by sampling rankings according to a Mallows model, for which we obtain pseudo-Euclidean embeddings to use with our tensor decomposition estimator.

In Figure 6.3 (top left), we show that as we increase the number of samples, we can obtain an increasingly accurate estimate of $\mathbf{T}$—exhibiting the *nearly-consistent* behavior predicted by our theoretical claims. This leads to downstream improvements in parameter estimates, which also become more accurate as n increases. In contrast, we find that the estimates of the same parameters given by Shin et al. (2022a) do not improve substantially as n increases, and are ultimately worse (see Figure 6.3, top right). Finally, this leads to improvements in the label model accuracy as compared to that of Shin et al. (2022a), and translates to improved accuracy of an end model trained using synthetic samples (see Figure 6.3, bottom).

Riemannian manifolds: Hyperbolic regression We similarly evaluate our estimator using synthetic labels from a hyperbolic manifold, matching the setting of Section 6.5. As shown in Figure 6.4, we find that our estimator consistently outperforms that of (Shin et al., 2022a), often by an order of magnitude.

6.7 Conclusions

We studied the theoretical properties of weak supervision applied to structured prediction in two general scenarios: label spaces that are finite metric spaces or constant-curvature manifolds. We introduced ways to estimate the noise rates of labeling functions, achieving consistency or near-consistency. Using these tools, we established that with suitable modifications downstream structured prediction models maintain generalization guarantees. Future directions include extending these results to even more general manifolds and removing some of the assumptions needed in our analysis.

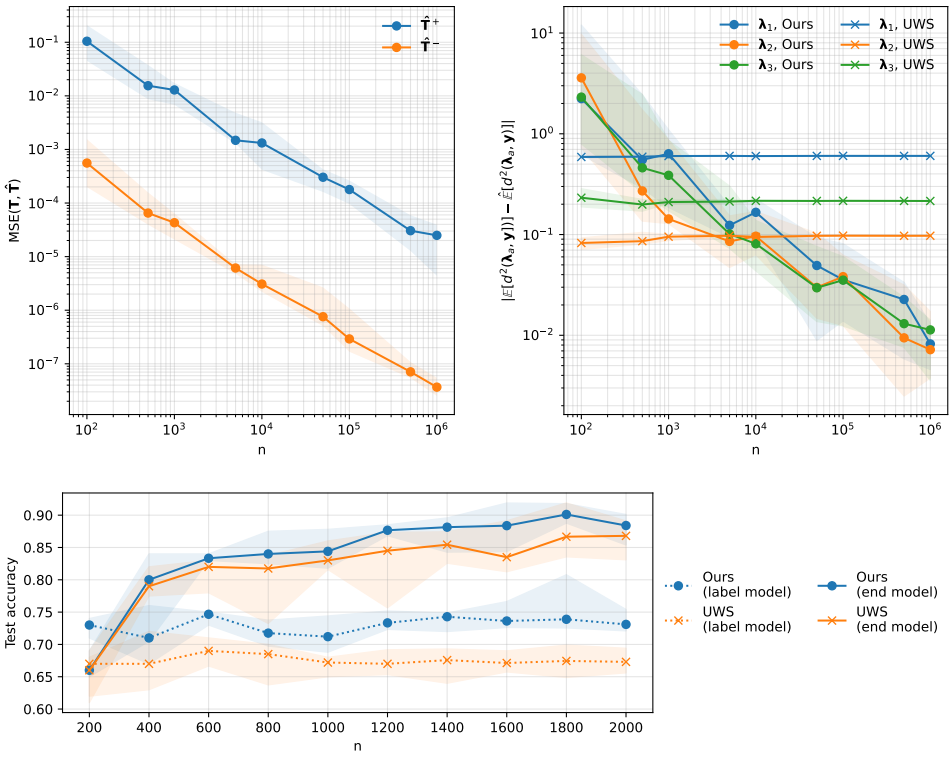


Figure 6.3: **Finite metric space case.** Parameter estimation improves with samples n in learning to rank—showing nearly-consistent behavior. Our tensor decomposition estimator outperforms (Shin et al., 2022a). In particular, (top left) as the number of samples increases, our estimates of the positive and negative components of $\mathbf{T}$ improve. (Top right) the improvements in $\mathbf{T}$ recovery with more samples translates to significantly improved performance over (Shin et al., 2022a), which is close to constant across n. (Bottom) this improved parameter estimation further translates to improvements in label model accuracy (using only the noisy estimates for prediction, without training an end model) and end model generalization. For the top two plots, we use $\boldsymbol{\theta} = [6, 3, 8]$, and in the bottom plot, we use $\boldsymbol{\theta} = [0, 0, 1]$. In all plots, we report medians along with upper and lower quartiles across 10 trials.

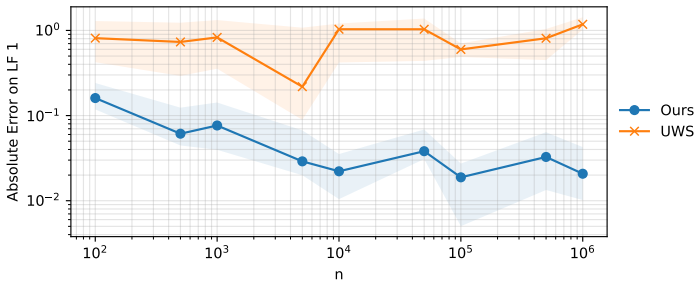


Figure 6.4: **Continuous case.** Parameter estimation improves with more samples in the hyperbolic regression problem. Our estimator outperforms Shin et al. (2022a). Here, we use different randomly sampled values of $\boldsymbol{\theta}$ for each run. We report medians along with upper and lower quartiles across 10 trials.

Part II

Improving Reliability and Accuracy at Inference-time

Chapter 7

Overview

7.1 Motivation

The ultimate utility of a machine learning model lies in its performance when deployed in real-world settings. Deployment introduces challenges that go beyond standard training and evaluation pipelines. Models often encounter inputs that differ significantly from their training distribution, and failure on such out-of-distribution or ambiguous inputs can have serious consequences—particularly in high-stakes domains like healthcare, finance, and law (Amodei et al., 2016).

Moreover, in many practical scenarios, retraining models to accommodate new data or environments could be prohibitively expensive, both computationally and logistically. This raises a central question: *How can we ensure reliable and accurate predictions at test-time without resorting to costly retraining?*

In this part of the thesis, we explore two complementary directions to address this challenge using statistical inference techniques:

(i) Develop methods that make models resilient to unforeseen and diverse inputs at deployment time.

(ii) Design principled techniques that allow models to improve their predictions during inference.

7.2 Literature

The problem of detecting out-of-distribution (OOD) samples is an instance of the broader challenge of detecting outliers or anomaly detection, where significant contributions have been made from several fields including data mining, statistics, and engineering (Rosner, 1975; Hawkins, 1980; Danuser and Stricker, 1998; Aggarwal and Yu, 2001; Chandola et al., 2009a,b). In this thesis, we are interested in ensuring that the models are robust to out-of-distribution (OOD) samples at test time. The popularity and widespread use of deep neural networks exposed their vulnerability to OOD samples (Nguyen et al., 2015b; Hendrycks and Gimpel, 2016). In response, a variety of solutions have been developed that are surveyed in detail by Salehi et al. (2022) and Yang et al. (2022b). The solutions include post-hoc techniques to quantify OOD uncertainty (Liang et al., 2018; Liu et al., 2020; Sun et al., 2021; Behpour et al., 2023; Ming et al., 2022), OOD-aware training-time regularization techniques (Hendrycks et al., 2019; Bevandić et al., 2018; Yang et al., 2021a; Katz-Samuels et al., 2022a), and approaches based on conformal prediction and hypothesis testing (Laxhammar and Falkman, 2011, 2015; Kaur et al., 2022; Bates et al., 2023). While this line of work has resulted in several interesting and powerful methods for uncertainty quantification (scoring functions), they are less effective in real-world deployments where *adaptivity* and strict adherence to *safety constraints* are paramount. A few recent methods adjust OOD scores at inference-time using distribution shifts observed during deployment (Zhang et al., 2022; Fan et al., 2022; Wang et al., 2020b). These works lack strict guarantees on safety constraints such as false positive rate (FPR) control, i.e., misclassifying OOD as in-distribution. The chapters 8 and 9 provide principled solutions for adaptive OOD detection while maintaining strict control on FPR. Our work builds on top of the existing methods for OOD uncertainty quantification and safe anytime-valid inference Darling and Robbins (1967). Classical work on anytime-valid confidence intervals (Darling and Robbins, 1967; Howard and Ramdas, 2022) provides guarantees over all time steps without requiring pre-specified stopping times. These sequences have been used in A/B testing (Johari, 2015) and bandits (Jamieson et al., 2014; Szorenyi et al., 2015). We adapt and extend this framework to handle importance-weighted samples in OOD detection, building on martingale-based constructions from Balsubramani (2015) to obtain confidence sequences valid

over both time and threshold choices.

In addition to being robust to OOD inputs, the models are expected to make reliable and accurate predictions on in-distribution (ID) inputs. Since the models at deployment are usually less than perfect (i.e., not 100% accurate), they may make mistakes on ID inputs. High-risk domains such as healthcare, finance, legal, etc., demand stricter control over such mistakes. Moreover, expensive retraining of models may not be feasible due to a lack of labeled data and computational costs. To mitigate these challenges, several solutions have been proposed. One line of work improves the predictions by equipping it with an option to abstain (El-Yaniv and Wiener, 2010; Mohri et al., 2024; Qiao and Valiant, 2019; Gangrade et al., 2021; Zhu and Nowak, 2022), or defer the predictions to human-in-the-loop (Madras et al., 2018; Tailor et al., 2024a; Verma and Nalisnick, 2022), or to more powerful models (Wolpert, 1992; DeSalvo et al., 2015; Jitkrittum et al., 2023; Gupta et al., 2024a), whenever it is uncertain about its predictions. Another vein focuses on inference techniques that can extract the most out of the model, possibly at the cost of additional compute cycles for inference. For instance, using test-time data augmentation (Shanmugam et al., 2021), zero-shot techniques (Xian et al., 2016; Wang et al., 2019; Shin et al., 2024b; Adila et al., 2024) have been effective in improving the performance. In the context of large language models (LLMs), which are often deployed in dynamic, open-ended environments, inference-time scaling (Snell et al., 2024; Chen et al., 2024; Zhang et al., 2025b). has emerged as a powerful tool to boost and improve performance. Methods like chain-of-thought reasoning (Wei et al., 2022a; Liu et al., 2023), and self-consistency (Wang et al., 2022b) aim to elicit better responses from the models at the cost of inference-time. In Chapter 10 of this thesis, we propose principled techniques based on conformal prediction (Vovk et al., 2005a) to improve the reliability and accuracy of LLMs in multiple-choice decision-making scenarios, common in tool-selection in agentic workflows (Qu et al., 2024).

7.3 Contributions

In this section, we provide a summary of our contributions towards improving the reliability and accuracy of models at inference-time. Specifically, we propose principled techniques to ensure robustness to out-of-distribution points at all times (Chapters 8, 9) and introduce conformal prediction based methods to improve LLMs' responses in multi-choice decision-making settings (Chapter 10).

7.3.1 Anytime Robustness to Out-Of-Distribution Points

ML models are typically trained on in-distribution (ID) data, with the expectation that at test-time the samples will come from the same ID distribution. However, when these models are deployed in the open world, they are subject to out-of-distribution (OOD) samples—points very different from the ID data. To ensure robustness to such inputs, prior works have focused on designing scoring functions to quantify OOD uncertainty and setting an appropriate threshold on these scores to achieve desired criteria such as 95% True Positive Rate (TPR), i.e., identifying In-Distribution (ID) points as ID (Yang et al., 2022a). Such an approach with a fixed scoring function and fixed threshold (FSFT) faces two issues, i) it lacks adaptivity to changing OOD distributions during deployment, and ii) it can lead to very high false positive rates (FPR), ranging from 60 to 96%, as observed in the Open-OOD benchmark (Yang et al., 2022a). In safety-critical real-life applications, e.g., medical diagnosis, controlling the False Positive Rate (FPR) is essential when dealing with various OOD samples dynamically. Motivated by these limitations, we develop principled methods for *adaptive* OOD detection with strict control over FPR at all times.

In Chapter 8, we present the first framework to adapt thresholds over time while providing strict control over the FPR. Setting appropriate thresholds for these scoring functions for OOD detection is challenging, as OOD samples (negative) are often unavailable up front. We consider settings where expert feedback is available if the system declares the point is OOD. For example, in medical applications, a radiologist's feedback is available. Utilizing this feedback, we propose a

novel mathematically grounded *Human-in-the-loop OOD detection framework* that incorporates expert human feedback to safely update the OOD detection threshold, ensuring robustness to variations in OOD data encountered after deployment. For settings with no shifts in distributions, we provide theoretical guarantees for our framework on controlling FPR at the desired level $\alpha \in (0, 1)$ at all times, and also provide a bound on the time taken to reach a given level of optimality. Using the insight from this analysis, we also propose an approach for settings with change points that reduces the duration of violation of FPR control. This framework integrates *expert feedback* and *safe anytime valid inference* (Balsubramani, 2015; Howard and Ramdas, 2022) to dynamically adjust the OOD detection threshold, thereby ensuring resilience to variations in OOD data encountered post-deployment. See Chapter 8 for further details.

The above framework can be used with any of the existing scoring functions proposed in the vast literature on OOD detection Yang et al. (2022a). However, we observed that while it can maintain FPR below α, the maximum TPR achievable is fundamentally limited by the choice of scoring function. Moreover, a fixed scoring function may not work well across a wide variety of OOD data encountered during deployment.

Motivated by these limitations, we extend the above framework in Chapter 9 to incorporate a principled mechanism for *learning the scoring function over time*. We formulate this as an optimization problem over a flexible space of scoring functions, with the objective of maximizing TPR while satisfying the FPR constraint. This adaptive scoring and thresholding framework jointly learns and updates the scoring function and the threshold based on feedback, enabling the system to respond flexibly to evolving OOD challenges while maintaining strict control over the false positives. This extension makes our approach significantly more robust and effective in open-world deployments.

7.3.2 Improving Language Model Inference with Conformal Prediction

Large language models (LLMs) have achieved impressive performance across a wide range of tasks, yet their reliability in decision-making scenarios, particularly in high-stakes or structured tasks like multiple-choice questions (MCQs) or tool usage, remains a challenge. In such settings,

even small mistakes can have outsized consequences. This makes it crucial to improve accuracy and quantify uncertainty in a principled way that can be used, for instance, to decide when to abstain from answering. Motivated by this need, in Chapter 10, we explore how conformal prediction (CP)—a statistical framework for reliable uncertainty estimation—can enhance LLMs' test-time performance.

A few recent works (Kumar et al., 2023; Su et al., 2024) applied conformal prediction with LLMs in the MCQ settings. In contrast to the ad-hoc approach of outputting a single choice (point prediction) with the highest score (e.g., softmax), CP, using the same scores, outputs a *prediction set*, a subset of answer choices that contains the true answer with high probability. This is referred to as the *coverage guarantee* of CP, and the size of the output sets quantifies the uncertainty of the model in the predictions—larger the sets higher the uncertainty. Our work in Chapter 10 utilizes and improves these prediction sets to improve the test-time performance of large language models (LLMs) in scenarios involving multiple-choice questions (MCQs) and tool usage.

First, we draw inspiration from human test-takers and the Monty Hall game (Rosenthal, 2008). In these scenarios, a reduction in answer choices improves the chances of answering correctly. We introduce *Conformal Revision of Question (CROQ)*, a method that applies conformal prediction (CP) to prune distractor choices in MCQs. This approach preserves the correct answer with high probability. The LLM is then prompted with the revised question with pruned choices to make a point prediction. Our empirical results demonstrate that predictions from CROQ are more accurate in comparison to the standard single round of inference.

The effectiveness of CROQ depends on generating small prediction sets, which in turn depends on the quality of the score function used in conformal prediction. Previous approaches have relied on potentially unreliable LLM logits or heuristic scoring methods. To address this limitation, we introduce CP-OPT, an optimization framework that learns score functions specifically designed to minimize prediction set sizes while maintaining the coverage guarantee. Our experiments show, CP-OPT produces smaller prediction sets than baseline methods using LLM logits at the same coverage level (95%). Moreover, we consider settings where the system decides to defer or abstain

based on the set size (uncertainty). Our results show that with smaller set sizes, CP-OPT also minimizes the number of deferrals (or abstentions). When CP-OPT is combined with CROQ, it delivers greater accuracy improvements than using CROQ with logit scores.

7.4 Background

7.4.1 Safe Anytime Valid Inference (SAVI)

Traditional statistical inference methods, such as confidence intervals and p-values, are typically valid only for fixed sample sizes. In contrast, many real-world applications—such as online A/B testing, bandits, and adaptive data analysis—require inference that remains valid as data arrive sequentially. This motivates the need for *safe anytime-valid inference* (SAVI), which allows for reliable statistical guarantees at arbitrary stopping times.

A core object in SAVI is the *confidence sequence*—a sequence of confidence intervals $\{\psi(t,\delta)\}_{t=1}^{\infty}$ such that,

$$\mathbb{P}(\forall t \geq 1 : |\theta - \hat{\theta}_t| \leq \psi(t,\delta)) \geq 1-\delta,$$

where θ is the true (but unknown) parameter of interest, such as the mean of the underlying data distribution, and $\hat{\theta}_t$ is the running estimate of θ at time t. Having such property ensures that the interval contains the true parameter at all times, providing strong guarantees even in adaptive settings.

The non-asymptotic Law of the Iterated Logarithm (LIL) provides such high-probability bounds that hold uniformly over time (Jamieson et al., 2014; Balsubramani, 2015; Schreuder et al., 2020; Howard and Ramdas, 2022). These bounds typically take the following form with constants $c_1, c_2, c_3 > 0$.

$$\psi(t,\delta) = \sqrt{\frac{c_1}{t}\Big(\log\log(c_2 t) + \log(\frac{c_3}{\delta})\Big)}$$

These non-asymptotic confidence intervals are valid simultaneously at all times and do not require

prior knowledge of the stopping time. These properties have made them valuable in applications like A/B testing (Johari et al., 2019), best-arm identification in bandits (Jamieson et al., 2014), etc.

Dvoretzky–Kiefer–Wolfowitz (DKW) Inequality. It is a foundational result in nonparametric statistics that provides uniform concentration bounds for the empirical cumulative distribution function (ECDF). Let $X_1, \ldots, X_n$ be i.i.d. samples from a real-valued distribution with cumulative distribution function (CDF) $F(x)$. The empirical CDF is defined as:

$$\hat{F}_n(x) = \frac{1}{n} \sum_{i=1}^{n} \mathbf{1}\{X_i \leq x\}.$$

The DKW inequality provides a finite-sample, distribution-free bound on the maximum deviation between $\hat{F}_n(x)$ and $F(x)$:

$$\mathbb{P}\left(\sup_{x \in \mathbb{R}} |\hat{F}_n(x) - F(x)| > \varepsilon \right) \leq 2e^{-2n\varepsilon^2}.$$

This result implies that, with probability at least $1 - \delta$, we have

$$\sup_{x \in \mathbb{R}} |\hat{F}_n(x) - F(x)| \leq \sqrt{\frac{1}{2n} \log\left(\frac{2}{\delta}\right)}.$$

This bound is uniform in x and holds for all distributions, making it particularly useful for constructing nonparametric confidence bands. The inequality was originally proven by Dvoretzky, Kiefer, and Wolfowitz in 1956 (Dvoretzky et al., 1956), and the tight constant 2 in the exponential term was established later by Massart (Massart, 1990a).

7.4.2 Conformal Inference

Conformal prediction (CP) is a framework for constructing prediction sets with rigorous statistical guarantees, assuming only that the data are exchangeable (Vovk et al., 2005b; Shafer and Vovk, 2008). In contrast to standard point predictions, CP outputs a set of predictions that contains the true label with a pre-specified probability, offering distribution-free uncertainty quantification.

Consider a dataset $\{(x_i, y_i)\}_{i=1}^n$ where (x_i, y_i) are i.i.d. samples from an unknown distribution. Given a new test input x_{n+1}, the goal of CP is to construct a prediction set $C(x_{n+1})$ such that

$$\mathbb{P}(y_{n+1} \in C(x_{n+1})) \geq 1 - \alpha,$$

for a chosen miscoverage rate $\alpha \in (0, 1)$. The key idea is to assign a *nonconformity score* to each example, which measures how unusual a data point is relative to the others. These scores are then used to calibrate predictions on a held-out calibration set.

In the standard split conformal prediction approach, the dataset is partitioned into a training set and a calibration set. A model f is trained on the training set, and a nonconformity function $s(x, y)$ (e.g., based on softmax probabilities or loss values) is computed on the calibration set to obtain scores $\{s_i = s(x_i, y_i)\}$. For a new input x_{n+1}, we compute scores $s(x_{n+1}, y)$ for all candidate labels y and define the prediction set as:

$$C(x_{n+1}) = \{y \in \mathcal{Y} : s(x_{n+1}, y) \leq q_{1-\alpha}\},$$

where $q_{1-\alpha}$ is the $(1 - \alpha)$ quantile of the calibration scores.

7.5 Outlook

As machine learning models, particularly large language models (LLMs), continue to grow in capability, their adoption across diverse and high-stakes domains is accelerating. This increasing reliance also underscores a critical issue: these models are complex, opaque, and prone to failure, especially when deployed in open-world environments. This thesis addresses the growing need to ensure *reliable and accurate inference-time behavior*, even in the face of OOD inputs. We demonstrate that statistical inference techniques such as *Safe Anytime Valid Inference (SAVI)* and *Conformal Prediction (CP)* can offer principled and practical solutions to this challenge.

Looking forward, several promising directions emerge from this work. Our human-in-the-loop framework for OOD detection assumes access to high-quality expert feedback. Several settings

may have noisy, delayed, or limited human feedback. An immediate future direction would be to adapt our approach to such settings with noisy feedback. To this end, the tools from weak supervision (Ratner et al., 2016) or noise-aware learning Natarajan et al. (2013a) could be useful. Moreover, the confidence sequences on the FPR estimate derived in our work are valid for a finite set of thresholds (quantiles). Due to the dependence introduced by importance sampling, the DKW-LIL type bounds (Howard and Ramdas, 2022), which are valid in i.i.d settings, were not directly applicable. Extending the validity of our confidence sequences to all quantiles under such importance sampling is an open problem.

Conformal prediction has been extensively applied in classification and regression settings (Angelopoulos and Bates, 2022) and remains relatively underexplored for language models. The enormous response space of LLMs makes ad-hoc application of CP impractical. Recent works have proposed *conformal factuality* frameworks (Mohri and Hashimoto, 2024; Cherian et al., 2024), aiming to improve the factuality of the LLM responses. Scores used to indicate the factual correctness of individual claims are crucial to the effectiveness of these techniques. Adapting the score learning framework presented in this thesis could be a fruitful direction. Further, the CROQ procedure in our work showed, the interactive nature of LLMs also opens new ways to use CP that go beyond the uncertainty quantification.

Chapter 8

Anytime Robustness to Out-Of-Distribution Points I

Robustness to out-of-distribution (OOD) samples is crucial for the safe deployment of machine learning models in the open world. Recent works have focused on designing scoring functions to quantify OOD uncertainty. Setting appropriate thresholds for these scoring functions for OOD detection is challenging as OOD samples are often unavailable up front. Typically, thresholds are set to achieve a desired true positive rate (TPR), e.g., 95% TPR. However, this can lead to very high false positive rates (FPR), ranging from 60 to 96%, as observed in the Open-OOD benchmark. In safety critical real-life applications, e.g., medical diagnosis, controlling the FPR is essential when dealing with various OOD samples dynamically. To address these challenges, we propose a mathematically grounded OOD detection framework that leverages expert feedback to *safely* update the threshold on the fly. We provide theoretical results showing that it is guaranteed to meet the FPR constraint at all times while minimizing the use of human feedback. Another key feature of our framework is that it can work with any scoring function for OOD uncertainty quantification. Empirical evaluation of our system on synthetic and benchmark OOD datasets shows that our method can maintain FPR at most 5% while maximizing TPR.

8.1 Introduction

Deploying machine learning (ML) models in the open world makes them subject to out-of-distribution (OOD) inputs: an ML model trained to classify on K classes also encounters points that do not belong to any of the classes in the training data. Modern ML models, in particular deep neural networks, can fail silently with high confidence on such OOD points (Nguyen et al., 2015b; Amodei et al., 2016). Such failures can have serious consequences in high-risk applications, e.g., medical diagnosis and autonomous driving. Safe deployment of ML models in an open world setting needs mechanisms that ensure robustness to OOD inputs. The importance of this problem has led to the development of many methods for OOD detection (Liang et al., 2017; Lee et al., 2018a; Liu et al., 2020; Ming et al., 2022) which aim to produce a *score* that can be used to decide on OOD vs in-distribution (ID) for a given point. For a detailed survey of literature in the area of generalized OOD detection, see Yang et al. (2021b).

ID data is usually plentiful, but we do not get to see different kinds of OOD samples before deployment. Consequently, many works in OOD detection are largely limited to *static settings* where the ID data is used to set a threshold on the scores used for detection (Liang et al., 2017; Liu et al., 2020; Ming et al., 2022). In these scenarios, this is usually done by setting a threshold that achieves a certain level of true positive rate (TPR), such as 95%. However, this can lead to a very high false positive rate (FPR), e.g., ranging between 60% to 96% as observed in the Open-OOD benchmark (Yang et al., 2022a). Furthermore, even if the ID data distribution remains the same after deployment, the OOD data could vary, resulting in highly fluctuating FPR. Thus, having a small, fixed amount of OOD data collected a priori to validate the FPR at a given threshold would not help in guaranteeing the desired FPR.

In safety-critical applications, the consequences of classifying an OOD point as ID (false positive) are more catastrophic than classifying an ID point as OOD (false negative). For example, in the medical diagnosis of brain scans, when the system is in doubt, it is better to classify a scan as OOD and defer the decision to human experts rather than for the ML model to give it an ID label i.e., predicting an in-distribution disease or classifying it as a normal scan. Therefore, safely

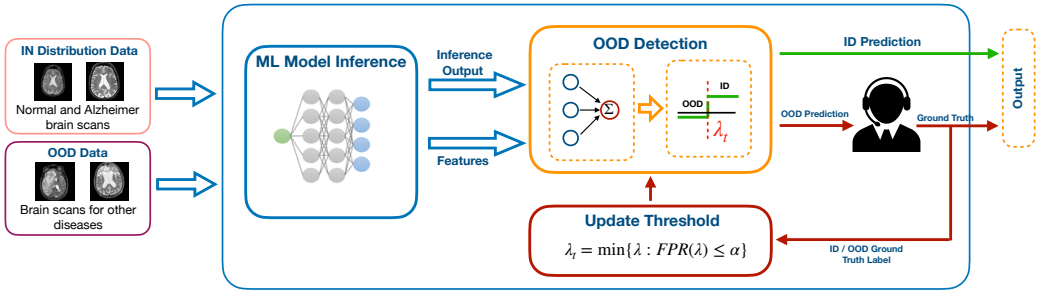


Figure 8.1: Illustration of OOD detection with human-in-the-loop with FPR control. In this example, the ID data is of brain scans of normal people and those with Alzheimer's disease. The OOD data could be anything other than these, e.g., brain scans of patients with some other diseases.

using ML models in such applications requires systems guaranteeing that the FPR is below a certain acceptable rate, e.g., FPR below 5%.

Furthermore, it is difficult to anticipate or collect the exact type of OOD data that the system can encounter during deployment. Thus, it is crucial that such systems adapt to the OOD data while controlling the FPR. Motivated by these challenges, we pose the following goal for safe OOD detection.

Goal: Develop a human-in-the-loop out-of-distribution detection system that has guaranteed false positive rate control while minimizing the amount of human intervention needed.

Our Contributions. Toward this goal, we make the following contributions:

1. **Human-in-the-loop OOD detection framework.** We propose a novel mathematically grounded framework that incorporates expert human feedback to safely update the OOD detection threshold, ensuring robustness to variations in OOD data encountered after deployment. Our framework can be used with any scoring function.
2. **Guaranteed FPR control.** For stationary settings, we provide theoretical guarantees for our framework on controlling FPR at the desired level at all times and also provide a bound on the time taken to reach a given level of optimality. Using the insight from this analysis, we also propose an approach for settings with change points that reduces the duration of violation of FPR control.

3. **Empirical validation on benchmark datasets.** We evaluate our framework through extensive simulations both in stationary and distribution shift settings. Through experiments on benchmark OOD datasets in image classification tasks with various scoring functions, we demonstrate the effectiveness of our proposed framework.

We emphasize that our aim is to develop a framework that can use any scoring function and safely adapt the threshold on the fly to enable the safe deployment of ML models. Therefore, our work is complementary to works that develop scoring functions for OOD detection.

8.2 Human-in-the-Loop OOD Detection

We propose a human-in-the-loop OOD detection system (Figure 8.1) that can work with any ML inference model and scoring function for OOD detection. We begin by describing the problem setting and then discuss each component of our proposed system in detail. See Algorithm 4 for step-by-step pseudocode.

8.2.1 Problem Setting

Data stream. Let $\mathcal{X} \subseteq \mathbb{R}^d$ denote the feature space and $\mathcal{Y} = \{-1, 1\}$ denote the label space for OOD detection with "1" denoting ID and "-1" denoting OOD. Let the distribution of ID and OOD data be denoted by $\mathcal{D}_{id}$ and $\mathcal{D}_{ood}$, respectively. Let $x_t \in \mathcal{X}$ denote the sample received at the time t. Let $y_t \in \{-1, 1\}$ denote the true label for x_t with respect to ID or OOD classification. We assume x_t are independent and drawn according to the following mixture model, $x_t \sim (1-\gamma)\ \mathcal{D}_{id} + \gamma\ \mathcal{D}_{ood}$, where $\gamma \in (0, 1)$ is the fraction of OOD points in the mixture. Note that $\mathcal{D}_{id}$, $\mathcal{D}_{ood}$ and γ are *unknown*.

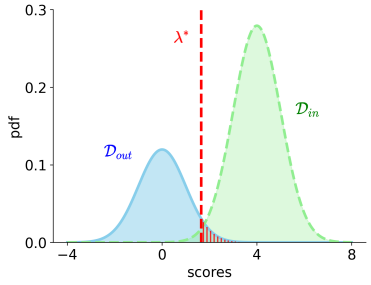


Figure 8.2: $\text{FPR}(\lambda^\star) = \alpha \longleftrightarrow \text{CDF}_{\mathcal{D}_{\text{out}}}(\lambda^\star) = 1 - \alpha$. Optimal $\lambda^\star$ for the optimization problem (P1) with $\alpha = 0.05$ and $x_t \overset{i.i.d}{\sim} 0.7\ \mathcal{D}_{in} + 0.3\ \mathcal{D}_{out}$, where $\mathcal{D}_{in}$ is $\mathcal{N}(4, 1)$ and $\mathcal{D}_{out}$ is $\mathcal{N}(0, 1)$.

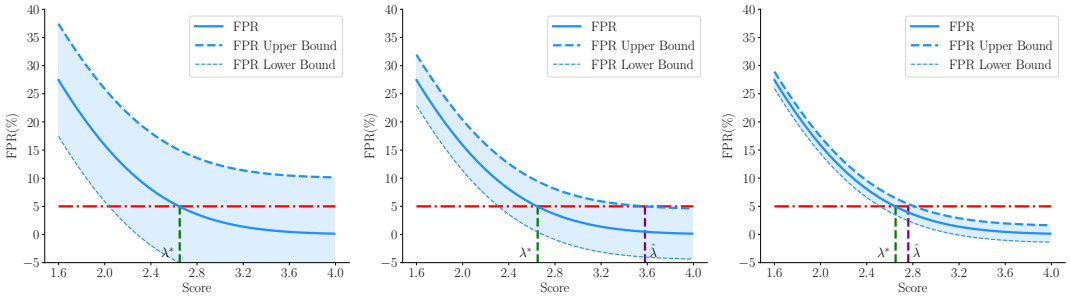
Scoring function. After receiving data point x_t, the system uses a given scoring function, $g : \mathcal{X} \mapsto \mathcal{S} \subseteq \mathbb{R}$, to compute a score quantifying the uncertainty of the point being ID or OOD. Our system is designed to work with any scoring function based OOD uncertainty quantification. Let $s_t = g(x_t)$ denote the score computed for point x_t. To be consistent across various scoring functions, let a higher score indicate ID and a lower score indicate OOD points. After computing score s_t the system needs to decide whether x_t is OOD or ID, it is done using a threshold-based classifier $h_\lambda : \mathbb{R} \mapsto \{-1, +1\}$ parameterized with $\lambda \in \Lambda \subseteq \mathbb{R}$: $h_\lambda(g(x)) = \text{sign}(g(x) - \lambda)$. Here we assume $\Lambda = (\Lambda_{\min}, \Lambda_{\max})$.

FPR and TPR. The population level FPR and TPR for any $\lambda \in \Lambda$ are defined as follows,

$$\text{FPR}(\lambda) = \mathbb{E}_{x \sim \mathcal{D}_{ood}}[\mathbf{1}\{g(x) > \lambda\}] \qquad \text{and} \qquad \text{TPR}(\lambda) = \mathbb{E}_{x \sim \mathcal{D}_{id}}[\mathbf{1}\{g(x) > \lambda\}].$$

Note that the cumulative distribution function (CDF) of $\mathcal{D}_{ood}$, $\text{CDF}_{\mathcal{D}_{ood}}(\lambda) = \mathbb{E}_{x \sim \mathcal{D}_{ood}}[\mathbf{1}\{g(x) \leq \lambda\}]$. Therefore, $\text{FPR}(\lambda) = 1 - \text{CDF}_{\mathcal{D}_{ood}}(\lambda)$. Similarly, $\text{TPR}(\lambda) = 1 - \text{CDF}_{\mathcal{D}_{id}}(\lambda)$. Since the CDF of any distribution is a monotonic function, both the FPR and TPR are monotonic in λ.

Expert human feedback. Our goal is to tackle critical applications where a human expert examines the samples that are declared as OOD (instead of the ML model making automatic predictions on them). The feedback obtained from the human expert can be used to safely update the OOD detection threshold at each time step, λ_t, so that the FPR is maintained below the desired rate of α. One can trivially control FPR by setting $\lambda_t = \Lambda_{\max}$, i.e., always getting human feedback. This would, of course, be too expensive and defeat the purpose of using an ML model in the first place. Therefore, in addition to controlling the FPR, we aim to minimize the human feedback solicited by the system. In an ideal system, the points declared as ID are directly classified by the ML model, and only the points declared as OOD are examined by a human expert. Thus, minimizing human feedback is equivalent to maximizing the TPR. This can be done by setting the threshold as, $\lambda_t := \arg\ \max_\lambda \text{TPR}(\lambda)$ subject to $\text{FPR}(\lambda) \leq \alpha$. Since the TPR is monotonic in λ, we can re-write this further as follows,



(a) No feasible solution, in the beginning (b) Feasible solution, after sometime (c) Near optimal solution, eventually

Figure 8.3: Illustration of the confidence interval defined in eq. (8.2) on FPR and their effect on threshold estimation. As the system receives more OOD samples, the confidence intervals will shrink and lead to better thresholds safely ($\hat{\lambda}_t \geq \lambda^\star$).

$$\lambda_t^\star := \arg\min_{\lambda \in \Lambda} \lambda, \quad \text{s.t.} \quad \text{FPR}(\lambda) \leq \alpha. \tag{P1}$$

The optimal threshold, denoted by $\lambda^\star$, is the smallest λ such that $\text{FPR}(\lambda^\star) = \alpha = 1 - \text{CDF}_{\mathcal{D}_{out}}(\lambda^\star)$ (see Figure 8.2). When the distribution of the OOD points, $\mathcal{D}_{ood}$, is not changing, then setting $\lambda_t^\star = \lambda^\star$ for all t would be the optimal solution. Note that changing the mixture ratio γ, or the distribution of the ID points $\mathcal{D}_{id}$ does not affect the value of the optimal threshold. As we do not have access to the true FPR and TPR values, we cannot solve the optimization problem (P1). Instead, we have to estimate the threshold at time t, denoted by $\hat{\lambda}_t$, using the observations until time t.

8.2.2 Adaptive Threshold Estimation

Ideally, we want to avoid human feedback for points that are determined as ID by the system, i.e., with a score greater than $\hat{\lambda}_t$. However, in order to have an unbiased estimate of the FPR and to detect potential changes in the distribution of OOD samples and therefore a change in true FPR, we obtain human feedback with a small probability p for points predicted as ID by the system. We refer to this as *importance sampling*.

FPR estimation and adapting the threshold. At each time t, we observe $x_t \overset{i.i.d}{\sim} (1-\gamma)\mathcal{D}_{id} + \gamma\mathcal{D}_{ood}$, and $s_t = g(x_t)$ is the corresponding score. If $s_t \leq \hat{\lambda}_{t-1}$, where $\hat{\lambda}_{t-1}$ is the threshold

determined in at time $t-1$, then it is considered an OOD point and hence gets a human label for it and we get to know whether it is in fact OOD or ID. If $s_t > \hat{\lambda}_{t-1}$, then x_t is considered an ID point and hence gets a human label only with probability p. So, we get to know whether it is truly ID or not with probability p. Now we have to update the threshold, $\hat{\lambda}_t$, such that the $\text{FPR}(\hat{\lambda}_t) \leq \alpha$, by finding the minimum λ that satisfies this constraint in order to maximize the TPR.

Our approach is based on using the feedback on the samples that are examined by human experts till time t to construct an unbiased estimator of $\text{FPR}(\lambda)$ (see Equation E.1). We also construct an upper confidence interval for the estimated $\text{FPR}(\lambda)$ that is valid at all thresholds $\lambda \in \Lambda$ and for all times simultaneously with high probability (see Equation 8.2). This enables us to optimize for λ such that the upper bound on the true $\text{FPR}(\lambda)$ is at most α at each time t and thus safely update the threshold λ_t. Let $S_t^{(o)} = \{s_1^{(o)}, \dots s_{N_t^{(o)}}^{(o)}\}$ denote the scores of the points that have been truly identified as OOD from human feedback so far, and $I_t^{(o)}$ be the corresponding time points. We estimate the FPR as follows,

$$\widehat{\text{FPR}}(\lambda, t) := \frac{1}{N_t^{(o)}} \sum_{u \in I_t^{(o)}} Z_u(\lambda), \tag{8.1}$$

$$Z_u(\lambda) := \begin{cases} \mathbf{1}(s_u^{(o)} > \lambda), & \text{if } s_u^{(o)} \leq \hat{\lambda}_{u-1} \\ \frac{1}{p}\mathbf{1}(s_u^{(o)} > \lambda), & \text{w.p. } p \text{ if } s_u^{(o)} > \hat{\lambda}_{u-1} \\ 0, & \text{w.p. } 1-p \text{ if } s_u^{(o)} > \hat{\lambda}_{u-1} \end{cases}.$$

We show that our estimator for the FPR is unbiased (the proof is deferred to the Appendix E.1.1).

Lemma 8.2.1. *Let $p > 0$, $\widehat{FPR}(\lambda, t)$ as defined in eq. (E.1) is an unbiased estimate of the true $FPR(\lambda)$, i.e., $\mathbb{E}[\widehat{FPR}(\lambda, t)] = FPR(\lambda)$.*

Algorithm 4: Human in the Loop OOD Detection

Input: FPR threshold α , sampling probability $p \in (0,1)$, Scoring function $g : \mathcal{X} \mapsto \mathbb{R}$,

$S_0 = \Phi$, $\hat{\lambda}_0 = \infty$
for $t = 1, 2, \ldots$ **do**
  Receive data point x_t ; $s_t = g(x_t)$
  if $s_t \leq \hat{\lambda}_{t-1}$ **then** $l_t = 1$
  else $l_t \sim \texttt{Bernoulli}(p)$
  end if
  if $l_t = 1$ **then**
    $y_t = \texttt{GetExpertLabel}(x_t)$
    $S_t = S_{t-1} \cup \{(s_t, y_t)\}$
  end if
  $\hat{\lambda}_t := \arg\min_{\lambda \in \Lambda} \lambda$ s.t. $\widehat{\text{FPR}}(\lambda, t) + \psi(t, \delta) \leq \alpha$
  if $l_t = 1$ **then** Output y_t
  else Output $\hat{y}_t = \text{sign}(s_t - \hat{\lambda}_t)$
  end if
end for

Finding threshold using a UCB on FPR. We propose using our estimated FPR with an upper confidence bound (UCB), which we will describe soon, to obtain the following optimization problem (P2),

$$\hat{\lambda}_t := \arg\min_{\lambda \in \Lambda} \lambda \quad \text{s.t.} \quad \widehat{\text{FPR}}(\lambda, t) + \psi(t, \delta) \leq \alpha, \tag{P2}$$

where the term $\psi(t, \delta)$ is a time-varying upper confidence which is simultaneously valid for all λ for all time with probability at least $1 - \delta$ for any given $\delta \in (0, 1)$. The minimization problem can be solved in many ways. We use a binary search procedure where we search over a grid on $(\Lambda_{\min}, \Lambda_{\max})$ with grid-size ν. The procedure searches for a smallest λ such that $\widehat{\text{FPR}}(\lambda, t) + \psi(t, \delta) \leq \alpha$. It uses eq. (E.1) to compute the empirical FPR at various thresholds and the confidence interval $\psi(t, \delta)$ given in eq. (8.2). Details of the binary search procedure are in the Appendix E.1.2.

Upper confidence bound (UCB). Our algorithm hinges on having confidence intervals on the FPR that are valid for all thresholds and for all times simultaneously. To construct such bounds, we use the confidence bounds based on Law of Iterated Logarithm(LIL) of (Khinchine, 1924).

We note that at each time step t, whether the sample x_t gets human feedback or not depends on the previous threshold $\hat{\lambda}_{t-1}$, which is a function of data up to time $t-1$ and the importance sampling. Therefore, *the samples used to estimate the FPR are dependent*, which prevents direct application of known results that are developed for i.i.d. samples (Howard and Ramdas, 2022). We build upon the LIL bounds for martingales (Balsubramani, 2015) and derive a confidence interval bound that is valid in our setting, which is given by the following equation,

$$\psi(t,\delta) := \sqrt{\frac{3c_t}{N_t^{(o)}}\left[2\log\log\left(\frac{3c_t N_t^{(o)}}{2}\right)+\log\left(\frac{2L}{\delta}\right)\right]}, \tag{8.2}$$

where $c_t = 1-\beta_t+\frac{\beta_t}{p^2}$, $\beta_t = \frac{N_t^{(o,p)}}{N_t^{(o)}}$ and $N_t^{(o,p)}$ is the number of points sampled using importance sampling until time t and $\nu \in (0,1)$ is a discretization parameter set by the user, $L = (\Lambda_{\max} - \Lambda_{\min})/\nu$.

8.3 Theoretical Guarantees

We want three provable properties for our approach. First, it must have guaranteed FPR control—the safety property we set out to ensure. In addition, we want to show a bound on the number of streamed observations (i.e., the time) taken to reach a point where every point does not need human feedback to ensure safety. Finally, we wish to have a notion of optimality, and a bound on the number of observations before it is reached. In this section, we provide a result that provides all of these properties under the following assumptions: (i) we are in the stationary setting, i.e., the distributions do not change over time, and (ii) the score distributions for the ID and OOD samples have sub-Gaussian tails. To quantify how close to the optimal operating point the system is at any given time, we define the following notion of η-optimality.

Definition 8.3.1. (η-optimality) For any $\eta > \text{FPR}(\lambda^*) - \lim_{\epsilon\to 0^+}\text{FPR}(\lambda^*+\epsilon)$, the system is said to be operating in the η-optimal regime after some time point T_η, if $\text{FPR}(\lambda^*) - \text{FPR}(\hat{\lambda}_t) \leq \eta$ for all $t \geq T_\eta$.

Using the estimated FPR in eq. (E.1) and the anytime valid confidence intervals on the FPR at

all thresholds we obtained in eq. (8.2), we provide the following guarantees for Algorithm 4.

Theorem 8.3.1. *Let $\alpha, \delta, p, \gamma \in (0,1)$. Let $x_t \overset{i.i.d}{\sim} (1-\gamma)\mathcal{D}_{id} + \gamma\mathcal{D}_{ood}$ and let $c_t = 1 - \beta_t + \frac{\beta_t}{p^2}$, $\beta_t = \frac{N_t^{(o,p)}}{N_t^{(o)}}$ where $N_t^{(o,p)}$ is the number of OOD points sampled using importance sampling until time t and $N_t^{(o)}$ is the total number of OOD points observed till time t. Let $n_0 = \min\{u : c_u N_u^{(o)} \geq 173\log(\frac{8}{\delta})\}$ and t_0 be such that $N_{t_0}^{(o)} \geq n_0$. If Algorithm 4 uses the optimization problem (P2) to find the thresholds with the upper confidence term $\psi(N_t^{(o)}, \delta/2)$ given by eq. (8.2), then there exist constants $C_1, C_2, C_3 > 0$ such that with probability at least $1-\delta$,*

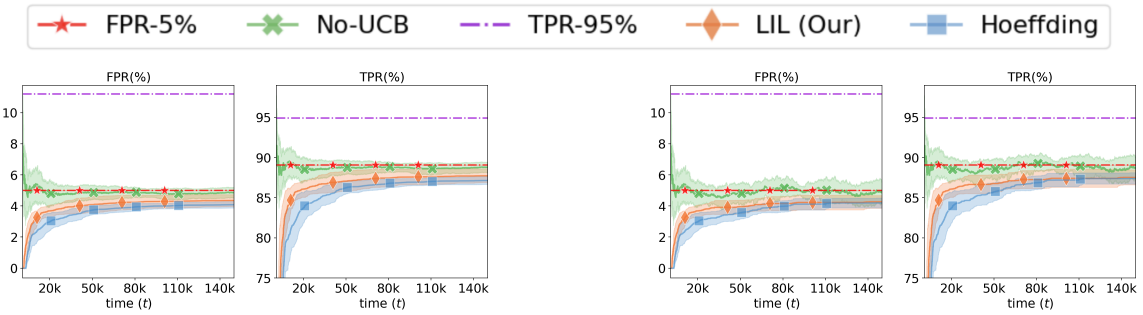
1. ***Controlled FPR.*** *For all $t \geq t_0$, $FPR(\hat{\lambda}_t) \leq \alpha$.*
2. ***Time to reach feasibility.*** *The algorithm will find a feasible threshold, $\hat{\lambda}_t$ such that $\widehat{FPR}(\hat{\lambda}_t) + \psi(N_t^{(o)}) \leq \alpha$, for all $t \geq \max(t_0, T_f)$, where, $T_f = \frac{2C_1}{\gamma\alpha^2}\log\left(\frac{4C_2}{\delta}\log(\frac{C_3}{\alpha})\right) + \frac{1}{\gamma^2}\log(\frac{4}{\delta})$.*
3. ***Time to reach $\eta-$optimality.*** *For all $t \geq \max(t_0, T_{\eta\text{-}opt})$, $\hat{\lambda}_t$ satisfy the η-optimality condition in definition 8.3.1, when $\widehat{FPR}(\hat{\lambda}_{T_{\eta\text{-}opt}}) \in [\alpha - \eta/2, \alpha]$ and $T_{\eta\text{-}opt} = \frac{8C_1}{\gamma\eta^2}\log\left(\frac{4C_2}{\delta}\log(\frac{2C_3}{\alpha})\right) + \frac{1}{\gamma^2}\log(\frac{4}{\delta})$.*

We discuss the results below and defer the proofs to the Appendix E.1.1.

Controlled false positive rate. We design our framework to safely update the threshold. Our method guarantees that $\lambda_t \geq \lambda^\star$ at all times, i.e., we approach the optimal $\lambda^\star$ from above and thus never violate the FPR constraint. This is crucial in applications where accurately controlling the FPR is essential.

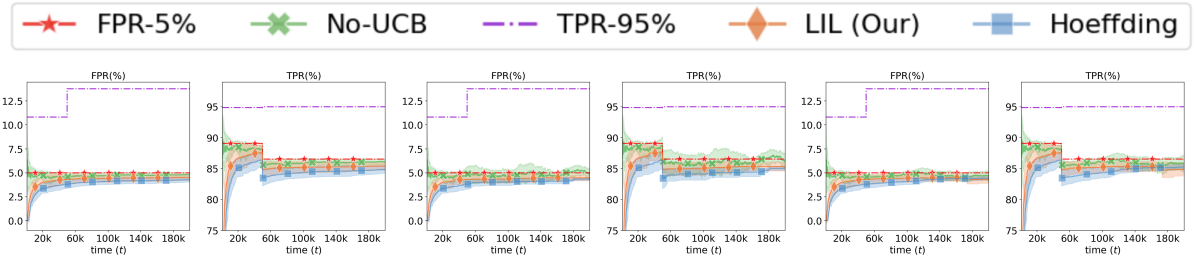
Time to reach feasibility. Our algorithm begins with setting $\lambda_0 = \Lambda_{\max}$, and therefore obtaining human feedback on all the points until the time point when we can find a threshold that enables us to safely declare scores above it as ID (Fig 8.3b). Our analysis provides an upper bound on the time taken by the algorithm to find such a safe, $\hat{\lambda}_t < \Lambda_{\max}$ such that, $\text{FPR}(\hat{\lambda}_t) \leq \alpha$. We call this time as *time to reach feasibility*, T_f, and it is the time step at which a sufficient number of observations $N_{T_f}^{(o)}$ is obtained so that the confidence interval $\psi(T_f, \delta/2) \leq \alpha$. It is inversely proportional to the level α and the fraction of OOD samples γ.

Time to reach η-optimality. As the time proceeds and the confidence intervals around the



(a) No window, no distribution shift. (b) 5K window, no distribution shift

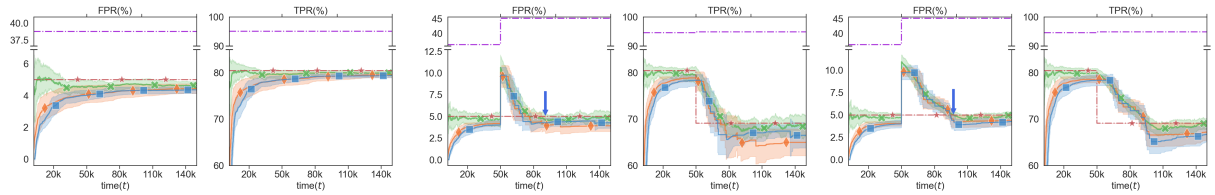
Figure 8.4: Results on the synthetic data with stationary distributions, $\gamma = 0.2$, and using no window. Each method is repeated 10 times. The mean and standard deviation are shown.



(a) Distribution shift, no window. (b) Distribution shift, 5k window. (c) Distribution shift, 10k window.

Figure 8.5: Effect of using various window sizes in synthetic data experiments. The distribution shift starts at $t = 50k$. The arrow indicates the time at which the mean FPR + std. deviation over 10 runs goes below 5% for the LIL method.

estimated FPR at different thresholds start to get smaller, the estimated safe threshold $\hat{\lambda}_t$ starts to approach $\lambda^\star$ (Fig 8.3c). If the estimated FPR at time step $T_{\eta\text{-opt}}$, denoted as $\widehat{\text{FPR}}(\hat{\lambda}_{T_{\eta\text{-opt}}})$, is within the range $[\alpha - \eta/2, \alpha]$ and the confidence interval $\psi(T_{\eta\text{-opt}}, \delta/2) \leq \eta/2$, then for all time points after $T_{\eta\text{-opt}}$ the algorithm will find a $\hat{\lambda}_t$ that satisfies the η-Optimality condition. In this regime, the algorithm operates in a state where the difference between the FPR at the true optimal threshold, $\text{FPR}(\lambda^*) = \alpha$, and the FPR at the estimated threshold $\text{FPR}(\hat{\lambda}_t)$, is bounded by η. Our analysis provides a bound on the time $T_{\eta\text{-opt}}$ which is the time point when the number of acquired OOD samples $N^{(o)}_{T_{\eta\text{-opt}}}$ becomes at least $\frac{4C_1}{\eta^2} \log\left(\frac{2C_2}{\delta} \log(\frac{2C_3}{\eta})\right)$. It is inversely proportional to the closeness to optimality η and the fraction of OOD samples γ.



(a) No distribution shift, no window. (b) Distribution shift, 5k window. (c) Distribution shift, 10k window.

Figure 8.6: Results with the KNN scores on CIFAR-10 as ID dataset. For (b) and (c) the distribution shifts at $t = 50k$. The arrow indicates the time at which the mean FPR + std. deviation over 10 runs goes below 5% for the LIL method.

8.4 Empirical Evaluation

We evaluate our method to verify the following claims:

C1. Compared to non-adaptive baselines, our approach achieves lower FPR while maximizing the TPR.

C2. In the stationary setting, our adaptive method based on the LIL upper confidence bound satisfies the FPR constraint at all times and produces high TPR.

C3. The proposed framework is compatible with any OOD scoring functions.

C4. Our method continues to work even in distribution shift settings with a simple adaptation using the windowed approach described in Section 8.4.2.

Baselines. We compare our method against the non-adaptive baseline popularly used for OOD detection. This non-adaptive method (**TPR-95**) finds a threshold achieving 95% TPR using the ID data and uses it at all times. For our adaptive method, we consider three choices of confidence intervals **i) No-UCB**: does not use any confidence intervals, **ii) LIL**: Uses confidence interval from eq. (LIL-Heuristic), and **iii) Hoeffding**: uses the confidence intervals from Hoeffding's inequality (Hoeffding, 1963). The confidence intervals from Hoeffding inequality are not valid simultaneously for all times, but are a reasonable choice for a practitioner.

$$\tilde{\psi}(t,\delta) = C_1 \sqrt{\frac{c_t}{N_t^{(o)}} \left(\log\log\left(C_2 c_t N_t^{(o)}\right) + \log\left(\frac{C_3}{\delta}\right)\right)}. \qquad \text{(LIL-Heuristic)}$$

The theoretical LIL bound in eq. (8.2) has constants that can be pessimistic in practice. We get around this by using a LIL-Heuristic bound which has the same form as in eq. (8.2) but with different constants. We consider the form in eq. (LIL-Heuristic). We find the constants C_1, C_2, C_3 using a simulation to estimate the bias of a coin with different constants and picking the ones so that the observed failure probability is below 5%. We use $C_1 = 0.5$ and $C_2 = 0.75$, $C_3 = 1$. We use $\alpha = 0.05$, $\delta = 0.2$, and importance sampling probability $p = 0.2$ through all the empirical evaluations. More details are available in the Appendix E.2.1, and code is provided[1].

Synthetic data setup. We simulate the OOD and ID scores using a mixture of two Gaussians $\mathcal{N}_{id}(\mu = 5.5, \sigma = 4)$ and $\mathcal{N}_{ood}(\mu = -6, \sigma = 4)$. We randomly draw 100k samples with $\gamma = 0.2$ (see Figure 8.4).

Real data setup. We use ID and OOD datasets and scoring functions from the OpenOOD benchmark (Yang et al., 2022a). Here we show the results on CIFAR-10 (Krizhevsky et al., 2009b) as an ID dataset and show the results on CIFAR-100, and Imagenet-1K (Deng et al., 2009) in the Appendix E.2. To verify C3, we use various scoring functions: ODIN (Liang et al., 2017), Mahalanobis Distance (Lee et al., 2018a), Energy Score (Liu et al., 2020), SSD (Sehwag et al., 2021), VIM (Wang et al., 2022c), and KNN (Sun et al., 2022) scores for the evaluation. We present results for the KNN (Sun et al., 2022) score here. For more details on the datasets, scores, and results on the rest of the scores, please see the Appendix E.2.

8.4.1 Stationary Distributions Setting

In the stationary setting the data distributions do not change over time. We use this setting to verify our theoretical claims as they are valid in such settings. We perform the experiments to verify claims C1 and C2 on synthetic and real data. See Figures 8.4(a) and 8.6(a) for the results. We make the following observations: (i)

We see that the non-adaptive method (TPR-95) with the fixed threshold has a high FPR at all times and violates the FPR constraint by a big margin. On the other hand, the adaptive methods

[1] https://github.com/2454511550Lin/TameFalsePositives-OOD

improve with time. (ii) We see that not using a UCB leads to violation of FPR constraints, and the methods with LIL-Heuristic, Hoeffding-based intervals are able to maintain the FPR below the user given threshold 5%. Moreover, all the methods improve as they acquire more samples with time and eventually reach very close to the optimal solution. We note that our method (LIL-heuristic) is faster in this regard than while maintaining safety.

Time to reach feasibility and optimality. In our theoretical results, we derived bounds on the time to reach feasibility (T_f) and the time to reach η-optimality, denoted by $T_{\eta-\text{opt}}$. These times are inversely proportional to the mixing ratio γ and the optimality level η. To verify this, we run the LIL method on the synthetic data setup with different values of γ and observe T_f (corresponding to $\alpha = 5\%$) and $T_{\eta-\text{opt}}$. We report the mean and std. deviation of T_f and $T_{\eta-\text{opt}}$ over 10 runs with different random seeds (see Table 8.1). We see both T_f and $T_{\eta-\text{opt}}$ decrease as γ increases and $T_{\eta-\text{opt}}$ is also inversely proportional to the optimality level. The corresponding FPR and TPR trends for each γ are shown in Figure 8.7. These trends also corroborate our understanding of the effect of γ on the time for feasibility and optimality.

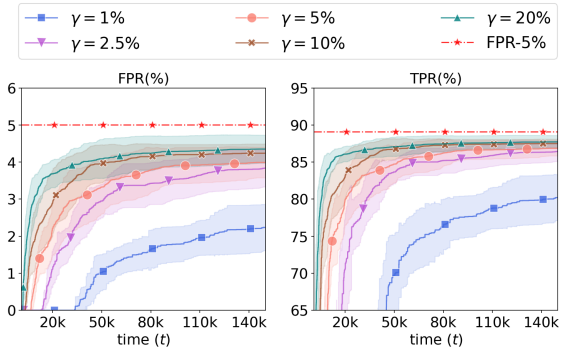


Figure 8.7: Results on the synthetic data with stationary distributions and different mixing ratios γ.

8.4.2 Distribution Shift Setting

We now proceed to investigate the case where the distributions change at a specific time point. One of the motivations for the proposed system is to be able to adapt to the variations of the OOD data. As long as $\mathcal{D}_{ood}$ does not change, any changes in the $\mathcal{D}_{id}$ or the mixing ratio γ do not affect the true FPR and therefore the optimal $\lambda^\star$. However, the true FPR does get affected when $\mathcal{D}_{ood}$ changes. When there is a change in $\mathcal{D}_{ood}$, estimating the FPR using all the acquired samples so far heavily biases the estimate towards scores that are far behind in time from the previous $\mathcal{D}_{ood}$. This leads to a long delay before our unbiased estimate of FPR can catch up to

γ	T_f	$T_{\eta-\text{opt}}$			
		$\eta = 1.0\%$	$\eta = 1.5\%$	$\eta = 2.0\%$	$\eta = 2.5\%$
2.5%	14,167 ±602	93,011 ±27,387	71,089 ±25,654	70,559 ±35,056	37,534 ±9302
5%	7,054 ±301	53,971 ±20,816	47,143 ±24,004	39,864 ±22,328	32,473 ±20,262
10%	3,549 ±200	50,748 ±33,947	35,517 ±22,131	26,435 ±14,361	17,312 ±7,757
20%	1,770 ±72	40,240 ±37,751	28,943 ±31,138	9,004 ±3,383	6,500 ±2,495

Table 8.1: Time to reach feasibility T_f and optimality $T_{\eta-\text{opt}}$ in the stationary setting for different η and mixing ratios γ.

the change.

Windowed approach. To overcome this challenge, we propose a sliding window-based approach with adaptive methods. The user can set a window size $N_w > 0$, and the system will estimate the FPR and the confidence intervals using only the most recent N_w samples that are determined as OOD by human feedback. This allows the system to more quickly adapt the threshold that is well aligned with the new distribution(s) of OOD samples.

Change detection. We use the following criteria to detect change, if $\widehat{\text{FPR}}(\hat{\lambda}_{t-1}, t) - \psi(t, \delta) > \alpha$ then it means the OOD distribution has changed. Here, the $\widehat{\text{FPR}}$ and ψ are computed using the samples in the window. We use change detection only for the methods with confidence intervals. The window size N_w has trade-offs, i.e., using a smaller window will enable faster change detection and adapting to the new distribution, but imposes limitations on the optimality as the smallest width of the confidence interval possible is inversely proportional to the window size. We verify claim C4 and study these trade-offs using experiments on synthetic and real data.

For the synthetic data, we use the same ID and OOD distribution as above till $t = 50k$ and change the OOD distribution to $\mathcal{n}_{ood}(\mu = -5, \sigma = 4)$ at time $t = 50k$. In the real data setting, we use the CIFAR-10 as ID and a mixture of MNIST, SVHN, and Texture datasets as OOD till $t = 50k$ and a mixture of TinyImageNet, Places365, and CIFAR-100 as OOD after $t = 50k$. We run TPR-95, and adaptive methods LIL, and Hoeffding in these settings with different choices of

window sizes, with 10 repeated runs using different random seeds, and show the mean and std. deviations of the FPR and TPR in Figures 8.5 and 8.6. We find that the windowed approach adapts more quickly compared to the method without a window (see Figures 8.5(a),8.5(c)).

Effect of window size. The results with various window sizes are shown in Figures 8.5(b), 8.5(c) on synthetic data and in Figures 8.6(b), 8.6(c) on real data. We also show the box plots of change detection times with LIL in Figure 8.8. As expected, we see that with a smaller window size, the change is detected earlier and the method is able to adapt faster. We also see that while a smaller window helps in faster adaptation, but limits how close to the optimal TPR is achieved.

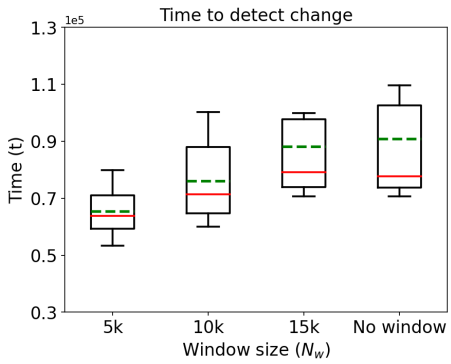


Figure 8.8: Box plot of change detection times with different window sizes on synthetic data. The median and mean are shown using solid and dashed lines, respectively.

To showcase the effect of **window-based approach in a stationary setting**, we run the methods with a window size of 5k in the fixed distribution setting (see Figure 8.4(b)). We observe similar behavior to the case without a window but with higher variance as the confidence intervals are limited by the window size.

Restart after change detection. In the previous experiments, the algorithm kept using all the samples in its window even after detecting the change. The window can contain samples from the previous distribution for some time, which leads to prolonged violation of the FPR constraint. In safety-critical applications, one might apply a conservative approach, i.e., restart the algorithm after detecting the change by emptying the window and resetting the threshold to $\Lambda_{\max}$. We run the LIL-based method with this variation using different window sizes and show the FPR and TPR of a *median trend* in Figure 8.4.2. To get the median trend, we run the algorithm with 10 random seeds and pick the FPR and TPR trends corresponding to the run having the median change detection time. We see that the FPR and TPR drop to 0 immediately after the

change is detected, and then the method recovers gradually. The variation without a window took a longer time to detect the change, and hence lags behind the window-based methods in approaching optimality after the restart. With a 5K window, it detected the change earlier, but due to the small window, it is not able to reach optimality. The one with a window size of 10k is a good trade-off as it is neither too late in detecting the change nor lags too far in approaching the optimality.

8.5 Related Works

Recent years have seen significant advancements in OOD detection. Surveys by Salehi et al. (2022) and Yang et al. (2022b) compare outlier, anomaly, and novelty detection, framing generalized OOD detection. We explore related research in these domains.

Out-of-Distribution detection. Many recent works proposed methods to quantify a score (uncertainty) that can separate OOD and ID data points. These include confidence-based scores (Bendale and Boult, 2015; Hendrycks and Gimpel, 2017b), Temperature scaling as used in ODIN (Liang et al., 2018), Energy-based scores (Liu et al., 2020; Wang et al., 2021b; Wu et al., 2023). With-in the vein of post-hoc methods ReACT (Sun et al., 2021) and ASH (Djurisic et al., 2023) employ light-weight perturbations to activation functions at inference time. VIM (Wang et al., 2022c) generates a virtual OOD class logit and matches it with original logits using constant scaling. GradNorm (Huang et al., 2021) GradOrth (Behpour et al., 2023) use gradient information. Distance-based methods detect samples as OOD if they are relatively far away from ID data. These include minimum Mahalanobis distance from class centroids (Lee et al., 2018a; Ren et al., 2023a), distances between representations learned using self-supervised learning (Tack et al., 2020; Sehwag et al., 2021), non-parametric KNN distance (Sun et al., 2022) and distances on Hyperspherical embeddings (Ming et al., 2022). Density-based approaches employ probabilistic models to characterize the density of ID data, positing that OOD points should appear in regions of low density. Some of the notable works in this vein are (Bishop, 1994), Nalisnick et al. (2019) likelihood ratios (Ren et al., 2019), normalizing flows (Kirichenko et al., 2020), invertible networks

(Schirrmeister et al., 2020), input complexity (Serrà et al., 2020), confusion log probabilities (Winkens et al., 2020), likelihood regret scores for variational autoencoders (Xiao et al., 2020).

Training-time regularization addresses OOD detection by reducing the confidence or increasing the energy on the OOD points using auxiliary OOD data for model training (Bevandić et al., 2018; Geifman and El-Yaniv, 2019; Mohseni et al., 2020; Jeong and Kim, 2020; Wei et al., 2022b; Yang et al., 2021a; Lee et al., 2018b; Hendrycks et al., 2019; Katz-Samuels et al., 2022a). OOD data can be highly diverse and it is hard to collect or anticipate the kind of OOD data the model will see after deployment. Moreover, if there is some change in OOD data these methods will require expensive retraining of the model.

Controlled false OOD detection rate (FDR). PAC-style guarantees on the OOD detection aiming to minimize false detections of OOD points are provided in the supervised settings (Liu et al., 2018). There is an emerging line of works on OOD detection using Conformal Prediction (CP) (Vovk et al., 2005a). Various techniques use non-conformity measures (NCMs) to assess alignment with the training dataset. Examples of previously proposed NCMs include those based on random forests, ridge regression, support vector machines (Vovk et al., 2005a), k-nearest neighbors (Vovk et al., 2005a; Papernot and Mcdaniel, 2018), and VAE with SVDD (Cai and Koutsoukos, 2020). Conformal Anomaly Detection (ICAD) (Laxhammar and Falkman, 2011, 2015) combines statistical hypothesis testing, NCM scores and inductive conformal anomaly detection (ICAD) (Laxhammar and Falkman, 2015)for OOD detection. Building upon ICAD, the iDECODe method (Kaur et al., 2022) introduces a novel NCM measure for OOD detection. Bates et al. (2023) demonstrate issues with conformal p-values under the ICP framework, proposing a technique based on high-probability bounds to compute calibration-conditional conformal p-values.

We note the following key differences: (i) The definition of *inliers (ID) as positives* and *outliers (OOD) as negatives* in our work is the opposite of these works. Thus, controlling FPR in these works translates to guaranteeing a lower bound on TPR in our setting rather than controlling FPR (rate of predicting true OOD as ID). These works are for offline settings – fixed sets of

inliers and calibration set are used to learn the inference rule which is used to make predictions on a test set such that the fraction of true ID predicted as OOD remains below a desired rate. Our guarantees on FPR are valid for all time points ($\forall t > 0$), while the paper provides guarantees for detection in a given test set of points.

OOD detection with test-time optimization. MEMO (Zhang et al., 2022) uses multi-head models such that the trained model can be adapted with test time distribution shift. ETLT (Fan et al., 2022) train a separate linear regression model during test time to calibrate the OOD scores as OOD scores. Other works such as (Wang et al., 2020b) and (Iwasawa and Matsuo, 2021) address the issue in a post-hoc manner without altering the trained model. We consider these methods complementary to our work as our framework can adopt the calibrated OOD scores and adapt the threshold safely with FPR control.

Generalized OOD detection is a classical problem that has drawn several promising solutions from researchers across diverse fields including databases, networks, etc. For a comprehensive treatment of the topic, we refer the reader to the book on outlier analysis (Aggarwal, 2017) and recent surveys on ood detection (Yang et al., 2021b, 2022a; Salehi et al., 2022).

Time uniform confidence sequences also known as any time valid confidence sequences are confidence intervals designed for streaming data settings, providing time-uniform and non-asymptotic coverage guarantees (Darling and Robbins, 1967; Lai, 1976). Using these sequences gets rid of the requirement of selecting sample size (or stopping time) ahead of time. Due to these nice properties, they have been used in various applications including A/B Testing (Johari, 2015; Maharaj et al., 2023), Multi-armed bandits (Jamieson et al., 2014; Szorenyi et al., 2015). Iterated-logarithm confidence sequences for best-arm identification using mean estimation are utilized in (Jamieson et al., 2014; Kaufmann et al., 2016; Zhao et al., 2016). Similar to (Darling and Robbins, 1968), (Szorenyi et al., 2015) include a confidence sequence valid over quantiles and time, derived via a union bound applied to the DKW inequality (Dvoretzky et al., 1956; Massart, 1990b). (Howard and Ramdas, 2022) improve the upper bounds of (Szorenyi et al., 2015) by replacing the logarithmic factor with an iterated-logarithm one and improving the

constants. Similar to (Howard and Ramdas, 2022), in our setting, we require confidence sequences valid uniformly over all times and quantiles. However, their results are valid when samples are drawn i.i.d. and this assumption breaks in our setting due to importance sampling. Interestingly, despite this issue, we obtain a martingale sequence in our setting and build upon the work of (Balsubramani, 2015) to obtain confidence sequences valid uniformly over time and thresholds.

8.6 Conclusions

We presented a mathematically grounded framework for human-in-the-loop OOD detection. By incorporating expert feedback and utilizing confidence intervals based on the Law of Iterated Logarithm (LIL), our approach maintains control over FPR while maximizing the TPR. The empirical evaluations on synthetic data and image classification tasks demonstrate the effectiveness of our method in maintaining FPR at or below 5% while achieving high TPR. Our theoretical guarantees are valid for stationary settings. We leave the extension to non-stationary settings as future work.

Chapter 9

Anytime Robustness to Out-of-Distribution Points II

Machine Learning (ML) models are trained on in-distribution (ID) data but often encounter out-of-distribution (OOD) inputs during deployment—posing serious risks in safety-critical domains. Recent works have focused on designing scoring functions to quantify OOD uncertainty, with score thresholds typically set based solely on ID data to achieve a target true positive rate (TPR), since OOD data is limited before deployment. However, these TPR-based thresholds leave false positive rates (FPR) uncontrolled, often resulting in high FPRs where OOD points are misclassified as ID. Moreover, fixed scoring functions and thresholds lack the adaptivity needed to handle newly observed, evolving OOD inputs, leading to sub-optimal performance. To address these challenges, we propose a human-in-the-loop framework that *safely updates both scoring functions and thresholds on the fly* based on real-world OOD inputs. Our method maximizes TPR while strictly controlling FPR at all times, even as the system adapts over time. We provide theoretical guarantees for FPR control under stationary conditions and present extensive empirical evaluations on OpenOOD benchmarks to demonstrate that our approach outperforms existing methods by achieving higher TPRs while maintaining FPR control.

9.1 Introduction

Machine learning (ML) models are typically trained under the closed-world assumption—test-time data comes from the same distribution as the training data (in-distribution, or ID). However, during deployment, models often encounter out-of-distribution (OOD) inputs, such as data points that do not belong to any training class in classification. In safety-critical domains like medical diagnosis, it is crucial that the system refrains from generating predictions for OOD inputs. Instead, these inputs must be flagged so that corrective actions can be taken or human intervention sought. For this, an OOD detector is integrated into systems to monitor real-time samples and flag those that appear OOD, ensuring that unreliable predictions are not produced.

To this end, many OOD detection methods have been developed to distinguish OOD inputs from ID inputs (Yang et al., 2022a). While ID data is available during model training, diverse OOD examples remain unseen until deployment. This limited access to OOD data naturally leads to an ID-based detection approach: (1) a scoring function is designed or selected to quantify how likely an input is to be OOD (or ID) *using only ID data*, and (2) a threshold is set on these scores—*again based solely on ID data*—to achieve a desired true positive rate (TPR) (e.g., 95% of ID inputs are correctly identified as ID). While this approach has shown promising results, it suffers from the following issues:

1) High False Positive Rate (FPR). OOD detection systems are prone to false positives, where OOD inputs are misclassified as ID (Nguyen et al., 2015b; Amodei et al., 2016). In high-risk applications, false positives are often more critical than false negatives. For example, detecting non-existent cancer may simply trigger further analysis, whereas missing an actual case can be fatal. Thus, systems in such domains must have strict FPR control (e.g., below 5%). However, the common approach sets score thresholds solely using ID data to meet a target TPR, leaving FPR uncontrolled. As a result, FPRs can be very high—for instance, ranging from 32% to 91% when CIFAR-10 is used as ID in the OpenOOD benchmark (Yang et al., 2022a).

2) Lack of Adaptivity. During deployment, systems encounter diverse, unseen OOD inputs. However, no single scoring function works universally well for all types of OOD data. As a

result, the common approach of relying on a pre-designed scoring function and threshold risks being stuck with one that performs poorly on newly encountered OOD inputs—thereby limiting achievable TPR. These methods miss the opportunity to improve their scoring functions based on OOD inputs observed during deployment (Figure 9.1). In light of these limitations, real-world systems must *adapt their scoring functions and thresholds over time* to handle diverse OOD inputs, maximizing TPR while keeping FPR under control. These challenges—high FPR and lack of adaptivity—motivate the following goal.

Goal: Develop a human-in-the-loop OOD system that strictly controls FPR, adaptively maximizes TPR, and thus minimizes human intervention.

Toward this goal, we make the following contributions:

1. We introduce a framework that leverages human feedback to identify OOD inputs and *simultaneously update scoring functions and thresholds*, thereby maximizing TPR while ensuring strict FPR control throughout deployment (Figure 9.3).

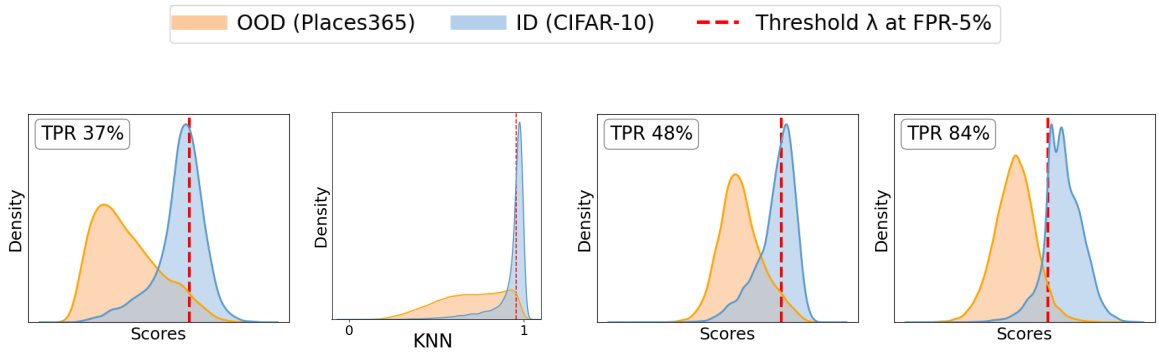
2. We provide a theoretical analysis that guarantees FPR control at all times, even as scoring functions and thresholds are updated. Our analysis shows that our approach can maintain FPR below a user-specified tolerance α (e.g., 5%) when the OOD remains constant.

3. We present extensive empirical evaluations under two different conditions: when the OOD remains the same (stationary) and when it changes over time (nonstationary). By comparing our method to baselines on OpenOOD benchmarks, we demonstrate that our framework consistently outperforms baselines, achieving higher TPR while ensuring strict FPR control.

We refer to our method as **ASAT** (*adaptive scoring, adaptive threshold*) in contrast to the common approach, denoted as **FSFT** (*fixed scoring, fixed threshold*), which uses fixed scoring functions and thresholds derived from ID data. We also compare with **FSAT** (*fixed scoring, adaptive threshold*),

Method	Score Fn	Threshold	FPR $\leq \alpha$	TPR
FSFT	Fixed	Fixed	×	Fixed (95%)
FSAT	Fixed	Adaptive	✓	Limited
ASAT	Adaptive	Adaptive	✓	Improved

Table 9.1: Comparison of scoring functions, thresholds, FPR control, and TPR for FSFT, FSAT, and ASAT.



(a) EBO (b) KNN (c) VIM (d) Optimal g^* from (P9.1)

Figure 9.1: Comparison of score distributions between post-hoc fixed scoring functions (trained solely on ID data) and our learned function g^* (trained offline on both ID and OOD). CIFAR-10 and Places365 are the ID and OOD datasets, respectively. As t increases, the ASAT-adapted scoring functions converge toward g^*, which achieves the highest TPR at 5% FPR compared to EBO, KNN, and VIM methods from the Open-OOD benchmarks. This comparison suggests that incorporating human-labeled OOD inputs encountered during deployment enables ASAT to update scoring functions that better align with real-world OOD points. For more details, see Section 9.5.

which updates thresholds while keeping scoring functions fixed. Please refer to Section 9.3.3 for descriptions of FSAT. See Table 9.1 for a high-level summary and comparison.

9.2 Preliminaries

9.2.1 Problem Setting

Data Stream. Let $\mathcal{X} \subseteq \mathbb{R}^d$ and $\mathcal{Y} = \{0, 1\}$ denote feature and label spaces, respectively, where 0 represents OOD and 1 represents ID. Let $\mathcal{D}_0$ and $\mathcal{D}_1$ be the *unknown* distributions of the OOD and ID data over $\mathcal{X}$. At each time t during deployment, our model sequentially receives a sample x_t drawn independently from the mixture model,

$$x_t \overset{\text{i.i.d.}}{\sim} \gamma\mathcal{D}_0 + (1-\gamma)\mathcal{D}_1,$$

where the mixture rate $\gamma \in (0, 1)$ is fixed but unknown. Let $y_t \in \mathcal{Y}$ be the true label for x_t, indicating whether it is an OOD or ID point.

Scoring Functions. Let $\mathcal{G} \subset \{g : \mathcal{X} \to \mathcal{S} \subseteq \mathbb{R}\}$ denote a class of scoring functions. For each

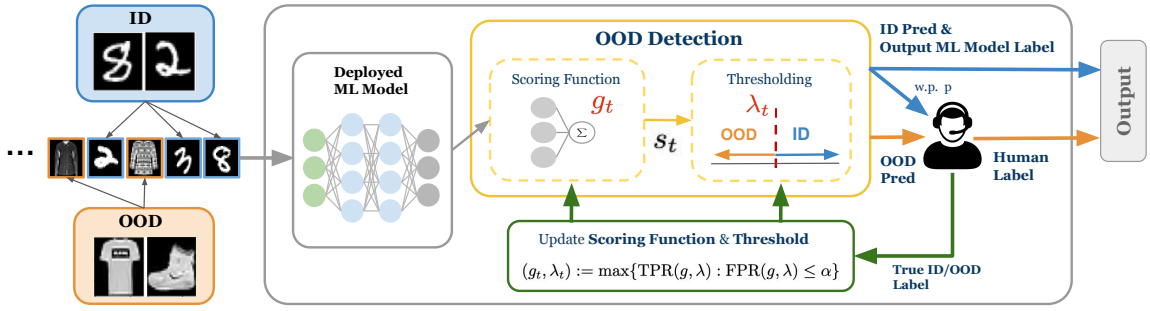


Figure 9.3: Overview of the ASAT workflow, updating both scoring function g_t and thresholds λ_t via human feedback.

x_t, a scoring function $g \in \mathcal{G}$ computes the score $s_t := g(x_t) \in \mathcal{S}$. We assume that higher scores correspond to a higher likelihood of x_t being ID. Then, the OOD classifier is defined by the threshold function $h_\lambda : \mathcal{S} \to \mathcal{Y}$ as $h_\lambda(s_t) := \mathbb{1}\{s_t > \lambda\}$, with $\lambda \in \Lambda$. The prediction label is $\widehat{y}_t := h_\lambda(s_t)$. The parameter λ defines the threshold for classifying x_t as either OOD or ID. Thus, setting λ carefully is crucial to ensure safety in OOD detection. In FSFT, both g and λ remain fixed for all $t \geq 1$. In contrast, ASAT aims to update both g_t and λ_t using incoming OOD inputs, thereby better controlling FPR and maximizing TPR.

Population-level FPR and TPR. Given any $g \in \mathcal{G}$ and $\lambda \in \Lambda$, the *true* FPR and TPR are defined as

$$\begin{aligned} \text{FPR}(g,\lambda) &:= \mathbb{E}_{x\sim\mathcal{D}_0}[\mathbb{1}\{g(x) > \lambda\}], \\ \text{TPR}(g,\lambda) &:= \mathbb{E}_{x\sim\mathcal{D}_1}[\mathbb{1}\{g(x) > \lambda\}]. \end{aligned} \tag{9.1}$$

Given g, FPR and TPR are monotonically decreasing in λ (Figure 9.2).

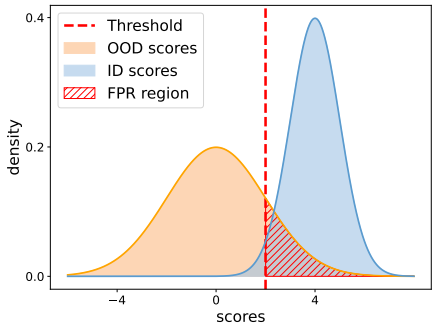


Figure 9.2: An example of OOD and ID score distributions mapped by g with a decision threshold λ. The hatched region is the FPR density.

9.3 Methodology

We propose ASAT (Figure 9.3) that adaptively updates both scoring functions and thresholds based on human-labeled OOD samples observed during deployment. This adaptation is crucial, as the initial scoring function and threshold—typically developed solely on ID data—may lead to

high FPR or suboptimal performance on certain OOD cases.

9.3.1 ASAT Workflow

At each time t, the system processes an input x_t using scoring function $\widehat{g}_{t-1}$ and threshold $\widehat{\lambda}_{t-1}$, both learned at the previous time $t-1$ (see Section 9.3.2 for the learning process). It first computes score $s_t = \widehat{g}_{t-1}(x_t)$ for x_t. If $s_t < \widehat{\lambda}_{t-1}$, then x_t is considered OOD and receives a human label $y_t \in \{0, 1\}$. If $s_t \geq \widehat{\lambda}_{t-1}$, x_t is considered ID, and in this case, we obtain the human label y_t only with probability $p \in (0, 1)$. We refer to this selective sampling as *importance sampling*. This importance sampling enables the detection of OOD shifts and allows for unbiased FPR estimation, which is essential to maintain FPR control.

9.3.2 Learning Scoring Functions and Thresholds

In this section, we describe how ASAT learns scoring functions and thresholds at each time t. We first define the optimal scoring function and threshold based on the TPR and FPR (9.1), which are unknown in practice. We then present a tractable and practical learning method only using finite samples and estimates.

Optimal Scoring Function and Threshold. ASAT aims to maintain strict FPR control during deployment while minimizing human intervention. In practice, accurately predicting IDs reduces the need for human verification. Hence, maximizing TPR effectively minimizes human intervention. With this, we formulate the ASAT objective as a joint optimization problem to find the optimal $(g^\star, \lambda^\star)$ that maximizes TPR under the constraint FPR $\leq \alpha$.

$$g^\star, \lambda^\star := \underset{g \in \mathcal{G}, \lambda \in \Lambda}{\arg\max} \quad \mathrm{TPR}(g, \lambda) \quad \text{s.t.} \quad \mathrm{FPR}(g, \lambda) \leq \alpha. \tag{P9.1}$$

Here, α is the user-specified tolerance level for FPR (e.g., $\alpha = 0.05$ implies FPR must be below 5% at all times). However, since $\mathcal{D}_0$ and $\mathcal{D}_1$ are unknown, the true values of TPR and FPR for a given (g, λ) are not accessible, making (P9.1) intractable. Consequently, we must estimate FPR and TPR based on the finite samples collected by time t.

FPR and TPR Estimates. We assume that ID data is largely available, whereas diverse OOD data is not accessible before deployment. Thus, ASAT identifies real-world OOD points via human feedback (i.e., importance sampling), while relying on a fixed, *i.i.d.* ID dataset from training. Let $X_t^{(ood)}$ denote the OOD samples collected up to time t (see Section 9.3.1), and let $X_0^{(id)}$ be the fixed ID dataset. Then, for a given scoring function and threshold pair (g, λ), the FPR estimate at time t is defined as

$$\widehat{\text{FPR}}_t(g, \lambda) := \frac{\sum_{u=1}^{t} Z_u \cdot \mathbb{1}\{g(x_u) > \lambda\}}{\sum_{u=1}^{t} Z_u}, \tag{9.2}$$

where

$$Z_u := \begin{cases} 1 & \text{if } s_u < \widehat{\lambda}_{u-1}, i_u = 0, y_u = 0, \\ \frac{1}{p} & \text{if } s_u \geq \widehat{\lambda}_{u-1}, i_u = 1, y_u = 0, \\ 0 & \text{otherwise,} \end{cases}$$

and i_u indicate whether x_u was importance sampled at time u. On the other hand, we estimate TPR using $X_0^{(id)}$ as

$$\widehat{\text{TPR}}(g, \lambda) := \frac{1}{|X_0^{(id)}|} \sum_{x \in X_0^{(id)}} \mathbb{1}\{g(x) > \lambda\}. \tag{9.3}$$

Note for a fixed g and λ the $\widehat{\text{TPR}}(g, \lambda)$ is constant over time, since $X_0^{(id)}$ is fixed.

Relaxations of (P9.1). Using the TPR and FPR estimates from eq (9.2) and (9.3), we can formulate a tractable optimization problem for the ASAT objective. However, because the indicator sums are non-differentiable and difficult to optimize, we further approximate them using a sigmoid function defined as $\sigma(\kappa, z) := 1/(1 + \exp(-\kappa z))$, $\kappa > 0$, which satisfies $\lim_{\kappa \to \infty} \sigma(\kappa, z) = \mathbb{1}\{z > 0\}$. This yields the following *smooth surrogate* estimates of the TPR and FPR:

$$\widetilde{\text{FPR}}_t(g, \lambda) := \frac{\sum_{u=1}^{t} Z_u \cdot \sigma(\kappa, g(x_u) - \lambda)}{\sum_{u=1}^{t} Z_u}, \tag{9.4}$$

$$\widetilde{\mathrm{TPR}}(g,\lambda) := \frac{1}{|X_0^{(id)}|} \sum_{x \in X_0^{(id)}} \sigma(\kappa, g(x) - \lambda). \tag{9.5}$$

The problem obtained by replacing the true FPR and TPR in (P9.1) with these surrogates is a non-convex constrained optimization problem. Although this type of problem has been widely studied (Park and Hentenryck, 2022; Jia and Grimmer, 2024), it remains challenging to solve efficiently in practice. Therefore, we reformulate it in a regularization form as follows.

$$\widehat{g}_t, \widehat{\lambda}'_t := \underset{g \in \mathcal{G}, \lambda \in \Lambda}{\arg\min} -\widetilde{\mathrm{TPR}}(g,\lambda) + \beta \cdot \widetilde{\mathrm{FPR}}_t(g,\lambda) \tag{P9.2}$$

The hyperparameter $\beta \in \mathbb{R}$ controls the trade-off between maximizing TPR and minimizing FPR. This formulation enables efficient gradient-based optimization of g and λ. With this, at each time t, ASAT estimates TPR and FPR using (9.4) and (9.5), then it solves the problem (P9.2). However, some issues remain to be addressed.

Periodically Learning Scoring Functions. Solving (P9.2) at every time t is computationally expensive, especially when only a few new OOD points have been identified. To address this, ASAT optimizes (P9.2) only periodically—when enough new OOD samples have been collected since the last update. Let U_t denote the number of updates to g performed by time t, and define ω_i^{ood} as the user-specified *optimization frequency* for $(i+1)$th update (e.g., ω_1^{ood} indicates the number of *new* OOD samples required for the second update). Define Δ_t^{ood} as the number of *new* OOD samples identified by time t *since the last update*. At time t, ASAT updates its scoring function only if $\Delta_t^{ood} \geq \omega_{U_t}^{ood}$.

Model Selection. Assume an update occurs at time t (i.e., $\Delta_t^{ood} \geq \omega_{U_t}^{ood}$) by solving (P9.2). However, the newly learned $\widehat{g}_t$ may underperform $\widehat{g}_{t-1}$ in terms of TPR at a given λ. To prevent this, we do a model selection by applying the TPR criterion:

$$\widehat{\mathrm{TPR}}(\widehat{g}_t, \widehat{\lambda}_t) + 2\zeta(\delta, |X_0^{(id)}|) > \widehat{\mathrm{TPR}}(\widehat{g}_{t-1}, \widehat{\lambda}_{t-1}).$$

Here, $\widehat{\lambda}_t$ is obtained from (Q1) (see Section 9.3.3), and $\zeta(\delta, |X_0^{(id)}|)$ is the Dvoretzky–Kiefer –Wolfowitz (DKW) confidence interval, valid w.p. at least $1-\delta$ (Nadaraya, 1965). If this is violated, we let $\widehat{g}_t = \widehat{g}_{t-1}$.

Threshold Estimation. The solution $(\widehat{g}_t, \widehat{\lambda}'_t)$ obtained from (P9.2) may violate the FPR constraint in (P9.1), since (P9.2) is unconstrained. That is, it could yield $\text{FPR}(\widehat{g}_t, \widehat{\lambda}'_t) > \alpha$, which undermines our goal of keeping FPR below α at all times in ASAT. To strictly enforce the FPR constraint, we introduce a new step: we estimate a *revised threshold* $\widehat{\lambda}_t$ for $\widehat{g}_t$ using the threshold estimation method proposed by Vishwakarma et al. (2024c).

9.3.3 Learning Thresholds

To ensure that the FPR constraint is satisfied throughout the deployment, we leverage the threshold estimation method FSAT (*fixed scoring, adaptive threshold*). See Table 9.1 for the differences between FSFT, FSAT, and ASAT.

FSAT for Threshold Estimation. The high-level idea of FSAT is to find a safe threshold for a *fixed* scoring function at each time t such that FPR (9.1) is below α. Given $(\widehat{g}_t, \widehat{\lambda}'_t)$ from (P9.2), we *discard* the potentially unsafe $\widehat{\lambda}'_t$ and compute a safe threshold $\widehat{\lambda}_t$ via the FPR estimate from eq (9.2). We achieve this by constructing a uniformly valid upper confidence bound (UCB) on the error between the estimated and true FPR.

Upper Confidence Bound (UCB). At each time t, the UCB is constructed as

$$\psi_t(\delta) := \sqrt{\frac{3c_t}{N_t^{(o)}}\left[2\log\log\left(\frac{3c_t N_t^{(o)}}{2}\right) + \log\left(\frac{2U_t|\Lambda|}{\delta}\right)\right]}, \tag{9.6}$$

where $\delta \in (0,1)$ is the failure probability, and c_t is a time-dependent variable. Intuitively, the UCB guarantees that w.p. at least $1-\delta$,

$$\widehat{\text{FPR}}_t(\widehat{g}_t, \lambda) \in \Big[\text{FPR}(\widehat{g}_t, \lambda) - \psi_t(\delta), \text{FPR}(\widehat{g}_t, \lambda) + \psi_t(\delta)\Big],$$

for all times $t \geq 1$ and thresholds $\lambda \in \Lambda$. That is, with high probability, the FPR estimate (9.2)

is within $\psi_t(\delta)$ of the true FPR. In Section 9.4, we provide further details, formal statements, and demonstrate that the UCB is valid for time-varying scoring functions. This leads to the optimization problem for adapting thresholds for a given $\widehat{g}_t$ in ASAT as

$$\widehat{\lambda}_t := \arg\min_{\lambda \in \Lambda} \quad \lambda \quad \text{s.t.} \quad \widehat{\mathrm{FPR}}_t(\widehat{g}_t, \lambda) + \psi_t(\delta) \leq \alpha \tag{Q1}$$

The problem (Q1) can be directly solved via binary search. Since TPR is monotonically decreasing in λ and $\widehat{g}_t$ is fixed, we reduce the objective from maximizing TPR to minimizing λ. Consequently, the solution $\widehat{\lambda}_t$ from (Q1), together with $\widehat{g}_t$, satisfies the FPR constraint with high probability, i.e., $\mathrm{FPR}(\widehat{g}_t, \widehat{\lambda}_t) \leq \alpha$ (see Proposition 1 for more details).

Below is an overview of ASAT. See Appendix F.2 for further details and complete algorithms.

ASAT Overview

For each time t,

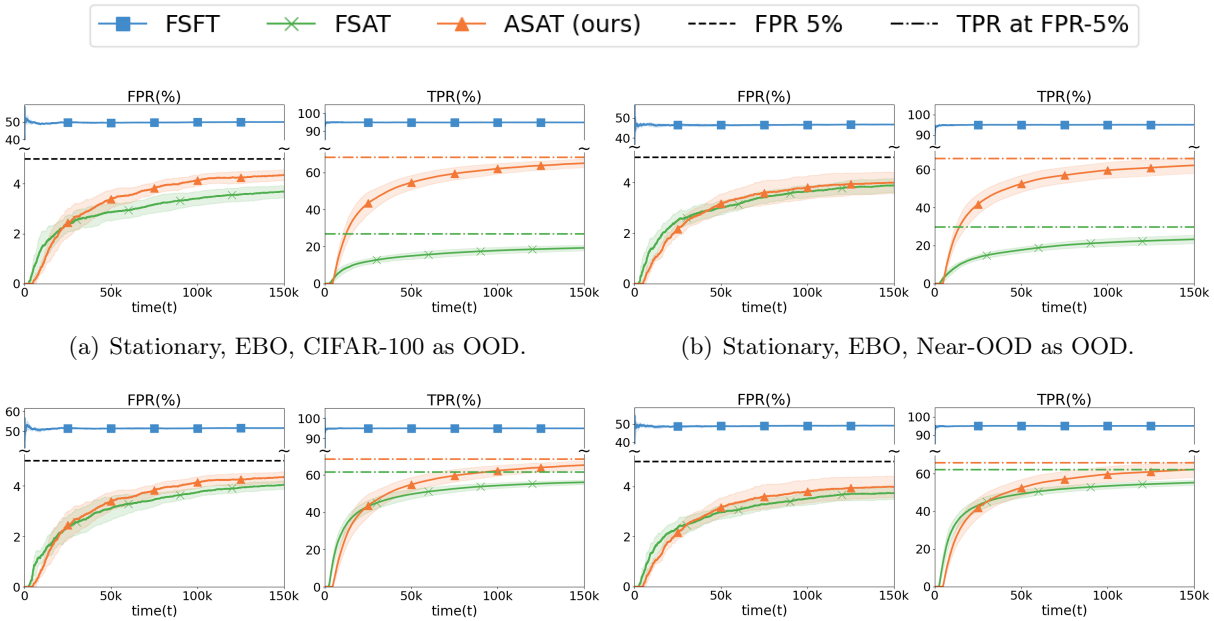
(Step 1) If $\Delta_t^{ood} > \omega_{U_t}^{ood}$, solve (P9.2) to obtain $\widehat{g}_t$ and $\widehat{\lambda}'_t$. Otherwise, let $\widehat{g}_t = \widehat{g}_{t-1}$.

(Step 2) Given $\widehat{g}_t$, discard $\widehat{\lambda}'_t$ and find $\widehat{\lambda}_t$ via (Q1).

9.4 Theoretical Analysis

We provide a theoretical analysis of FPR control in stationary settings, where both $\mathcal{D}_0$ and $\mathcal{D}_1$ remain fixed over time (but unknown). Our goal is to show that ASAT maintains $\mathrm{FPR}(\widehat{g}_t, \widehat{\lambda}_t) < \alpha$ at all times. The result is presented below.

Proposition 1. Let U_t be the total number of updates before time t. Let $N_t^{(o)} = \sum_{u=1}^{t} Z_u$, and $N_t^{(imp)}$ be the number of importance sampled OOD points. Define $\beta_t = N_t^{(imp)}/N_t^{(o)}$, $c_t = 1 + (1-p)\beta_t/p^2$, and $t_0 = \min\{u : c_u N_u^{(o)} \geq 173 \log(4/\delta)\}$. Let $\Lambda := \{\Lambda_{\min}, \Lambda_{\min} + \eta, \ldots, \Lambda_{\max}\}$, with discretization parameter $\eta > 0$. Then, w.p. at least $1 - \delta$, we have $\forall\, t \geq t_0$,



(a) Stationary, EBO, CIFAR-100 as OOD.

(b) Stationary, EBO, Near-OOD as OOD.

(c) Stationary, KNN, CIFAR-100 as OOD.

(d) Stationary, KNN, Near-OOD as OOD.

Figure 9.4: Results on the stationary setting for CIFAR-10 as ID, with each method repeated 5 times (mean and std shown). EBO and KNN are the initial scoring functions and are fixed for FSFT and FSAT. The dotted-dash lines represent the *expected* TPR at FPR-5% for EBO, KNN, and $\widehat{g}^*$ (matching colors).

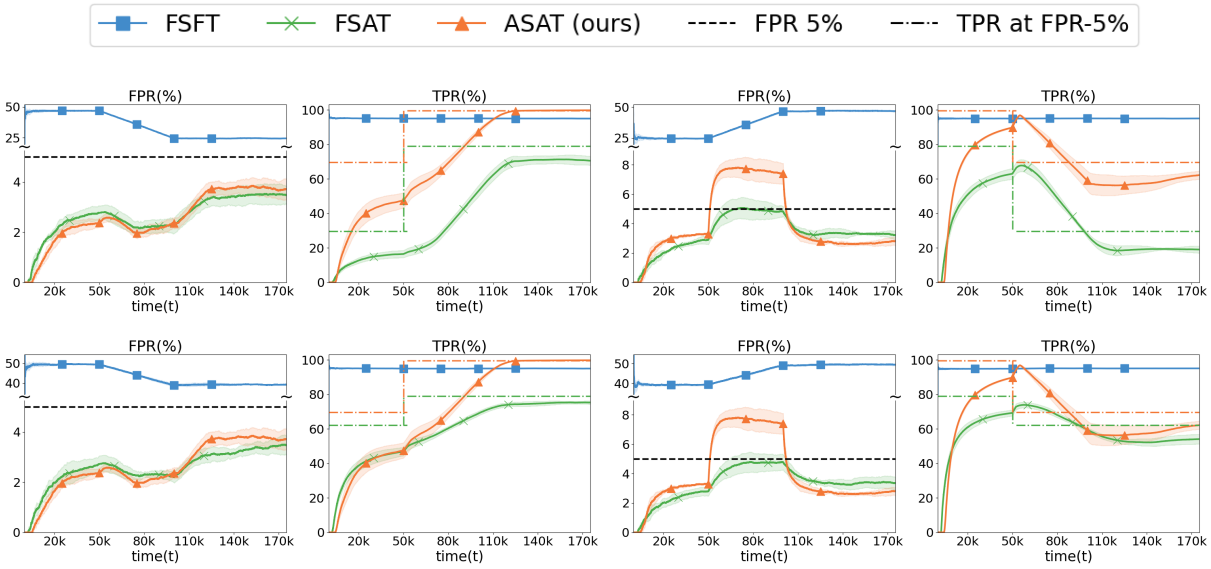
a) $\sup_{\lambda \in \Lambda} \left| \widehat{\mathrm{FPR}}_t(\widehat{g}_t, \lambda) - \mathrm{FPR}(\widehat{g}_t, \lambda) \right| \leq \psi_t(\delta)$, where

$$\psi_t(\delta) := \sqrt{\frac{3c_t}{N_t^{(o)}} \left[2 \log\log \left(\frac{3c_t N_t^{(o)}}{2} \right) + 2 \log \left(\frac{4U_t|\Lambda|}{\delta} \right) \right]}.$$

b) Consequently, $\mathrm{FPR}(\widehat{g}_t, \widehat{\lambda}_t) \leq \alpha$.

Interpretation. Proposition 1 establishes the desired FPR control at all times in ASAT. It ensures that w.p. at least $1 - \delta$, a) the FPR estimate (9.2) is within $\psi_t(\delta)$ of the true FPR for all t and λ, and consequently b) FPR $\leq \alpha$ in ASAT.

Discussion. Similar results were obtained for FSAT (which adapts thresholds for a *fixed* scoring function) using a similar UCB (Vishwakarma et al., 2024c). But, this approach does not extend directly to our setting, where ASAT also updates scoring functions and uses them for predictions and sample collections. Nevertheless, our design achieves comparable FPR control and higher TPR—since ASAT optimizes scoring functions to maximize TPR in (P9.2).

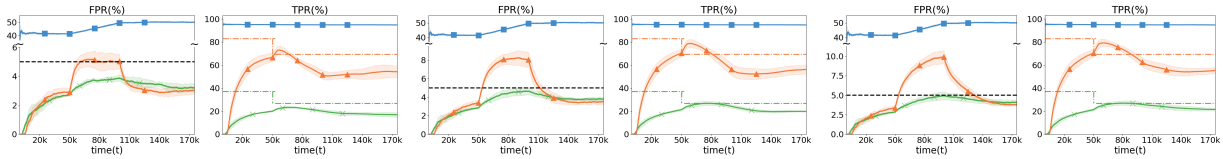


(c) KNN, 5k window, From Near-OOD to Far-OOD. (d) KNN, 5k window, From Far-OOD to Near-OOD.

Figure 9.5: Results on the nonstationary setting, with each method repeated 5 times (mean and std shown). A shift happens between Near-OOD and Far-OOD mixtures at time $t = 50$k. EBO and KNN are the initial scoring functions and are fixed for FSFT and FSAT. The dotted-dash lines represent the TPR at FPR-5% for EBO, KNN, and $\widehat{g}^*$ (matching colors).

However, these advantages come with a slight trade-off: $\psi_t(\delta)$ includes the additional U_t term to ensure that it is also valid for time-varying $\widehat{g}_t$. In practice, we anticipate $U_t \ll N_t^{(o)}$, since updates occur only periodically. Thus, this extra dependence on U_t results only in a minor increase in $\psi_t(\delta)$ from FSAT, which is offset by the improved TPR.

Proof Sketch. The proof of Proposition 1 entails challenges. First, the samples used to estimate the FPR in eq (9.2) are *dependent*. In particular, whether we receive the human label for x_t depends on the previously learned scoring function $\widehat{g}_{t-1}$ and threshold $\widehat{\lambda}_{t-1}$, both of which are functions of previous data $\{x_1, ..., x_{t-1}\}$. This dependency prevents the direct application of the *i.i.d.* LIL bound (Howard and Ramdas, 2022). To overcome this, we leverage the martingale structure of our data and apply the martingale LIL bound (Khinchine, 1924; Balsubramani, 2015), yielding a time-uniform confidence sequence. Additionally, we ensure that these confidence intervals hold simultaneously for all $\lambda \in \Lambda$ and all deployed scoring functions by applying the union bound. See Appendix F.3 for detailed proofs.



(a) EBO, 5k window. (b) EBO, 10k window. (c) EBO, 15k window

Figure 9.6: Results on the nonstationary setting with different window sizes 5k, 10k, and 15k, with each method repeated 5 times (mean and std shown). A shift happens from Places365 and CIFAR-100 at time $t = 50$k.

9.5 Experiments

We compare three methods: FSFT (fixed scoring functions and thresholds), FSAT (fixed scoring functions with adaptive thresholds), and our proposed ASAT (adaptive scoring functions and thresholds). We verify the following claims on OpenOOD benchmarks.

C1. In stationary settings, ASAT (ours) $\succ$ FSAT $\succ$ FSFT, where $a \succ b$ means that method a achieves better TPR than b while maintaining FPR below α, or that b violates the FPR constraint.

C2. ASAT adapts to nonstationary OOD settings, updating scoring functions and thresholds to maximize TPR with *minimal* FPR violations.

For completeness, we also compare ASAT against a naive binary classifier in Appendix F.4.1.

9.5.1 Experiment Settings

Datasets. We use CIFAR-10 (Krizhevsky et al., 2009b) as ID for the experiments. In stationary settings, we use CIFAR-100 and a mixture of CIFAR-100, Tiny-ImageNet (Deng et al., 2009), Places365 (Zhou et al., 2017) as OOD (referred to as *Near-OOD*). In nonstationary settings, we change the distribution at time $t = 50$k for all experiments. We use pairs of (1) CIFAR-100 and Places365 and (2) a mixture of MNIST (Deng, 2012), SVHN, and

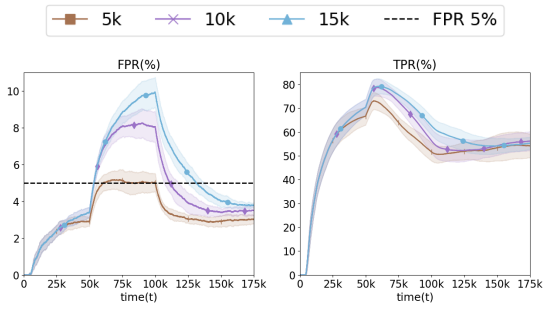


Figure 9.7: Results on nonstationary OOD from Places365 to CIFAR-100 with different window $N_w^{(est)}$; only ASAT results are shown.

Texture (referred to as *Far-OOD*) and a mixture of CIFAR-100, Tiny-ImageNet, and Places365 (Near-OOD).

See Appendix F.4.6 and F.4.7 for additional experiments on different feature extractor (ViT-B/16 (Dosovitskiy et al., 2021)), ID datasets (CIFAR-100, ImageNet-1k (Deng et al., 2009)), OOD datasets, and scoring functions.

Scoring Functions. We use post-processing scoring functions based on a ResNet18 model (He et al., 2015), which is trained on CIFAR-10, from Open-OOD benchmarks: ODIN (Liang et al., 2017), Energy Score (EBO) (Liu et al., 2020), KNN (Sun et al., 2022), Mahalanobis Distances (MDS) (Lee et al., 2018b), and VIM (Wang et al., 2022c). Both FSFT and FSAT used a fixed scoring function from the above benchmarks. For consistency, in ASAT, the same initial scoring function as FSFT is used until enough OOD samples are identified and the first update occurs.

Evaluation FPR and TPR. For evaluation, we count the actual number of false-positive and true-positive instances incurred by a method within a specific time frame. These *evaluation* FPR and TPR are defined as

$$\begin{aligned} \mathrm{FPR}_t^{(eval)} &:= \frac{\sum_{u=t'}^{t} \mathbb{1}\{y_u = 0, \widehat{y}_u = 1\}}{\sum_{u=t'}^{t} \mathbb{1}\{y_u = 0\}}, \\ \mathrm{TPR}_t^{(eval)} &:= \frac{\sum_{u=t'}^{t} \mathbb{1}\{y_u = 1, \widehat{y}_u = 1\}}{\sum_{u=t'}^{t} \mathbb{1}\{y_u = 1\}}, \end{aligned} \tag{9.7}$$

where $t' = t - N_w^{(eval)}$, and $N_w^{(eval)}$ denotes the window size of past instances used for evaluation.

Choices of $\mathcal{G}$. Our framework is flexible with the choice of $\mathcal{G}$. We use 2-layer ReLU neural networks that take a feature output from the penultimate layer of the Resnet18. To solve (P9.2), we use the Adam optimizer (Kingma and Ba, 2017). See Appendix F.4.3 for more details.

Hyperparameters and Constants. We perform a *grid search* to optimize our hyperparameters. In particular, we set $\beta = 1.5$ and $\kappa = 50$. For the constants, we use $\alpha = 0.05$, $\gamma = 0.2$, and $p = 0.2$ (See Appendix F.4.4).

Empirical Convergence to Optimality. We assess convergence of $(\widehat{g}_t, \widehat{\lambda}_t)$ by comparing $\mathrm{TPR}_t^{(eval)}$ (9.7) with the reference $\mathrm{TPR}(\widehat{g}^\star, \lambda)$. Here, $(\widehat{g}^\star, \lambda)$ is near-optimal for (P9.1), where

$\widehat{g}^\star$ is trained on *i.i.d.* OOD/ID data, and λ is chosen to satisfy $\text{FPR}(\widehat{g}^\star, \lambda) = \alpha$. Intuitively, $\text{TPR}(\widehat{g}^\star, \lambda)$ represents the best achievable TPR under FPR $\leq \alpha$, and we expect $\text{TPR}_t^{(eval)}$ to converge toward it as t increases in ASAT.

Thresholds for Baselines. While FSFT uses a fixed threshold at TPR-95%, both FSAT and ASAT update thresholds over time. For FSAT, we adopt the same heuristic UCB from Vishwakarma et al. (2024c). For ASAT we adapt the theoretical bound in (9.6) as

$$\psi_t^H(\delta) := c_1 \sqrt{\frac{c_t}{N_t^{(o)}} \left[\log\log\left(c_2 c_t N_t^{(o)}\right) + \log\left(\frac{c_3}{\delta}\right) \right]},$$

with $c_1 = 0.65$, $c_2 = 0.75$, and $c_3 = 1.0$. Here, c_2 and c_3 are adopted from Vishwakarma et al. (2024c), while c_1 is increased to account for the varying score functions (see Section 9.4). See Appendix F.4.2 for more details.

Stationary Setting. In this setting, $\mathcal{D}_0$ and $\mathcal{D}_1$ remain fixed. For evaluation, we set $N_w^{(eval)} = t$ (i.e., using all past data), since no abrupt changes in FPR/TPR are anticipated. For updating $\widehat{g}_t$, we set the optimization frequency $\omega_i^{(ood)}$ to 100 for the first 2k OOD points, 500 for the next 10k, and 1000 thereafter. This allows aggressive updates early on, with update frequency decreasing as $\widehat{g}_t$ becomes better trained.

Nonstationary Setting. We manually change the OOD $\mathcal{D}_0$ at $t = 50k$, since variations in γ or $\mathcal{D}_1$ do not impact the FPR. We set $N_w^{(eval)} = 50\text{k}$ for a more optimistic performance metric and use $\omega_i^{(ood)} = 100$ for the first 2k samples and 500 afterward to ensure $\widehat{g}_t$ adapts quickly to the new OOD. Since using all the past OOD samples (including *outdated ones*) becomes ineffective after a shift, we adopt a *windowed approach* that relies only on the most recent $N_w^{(est)}$ OOD samples for FPR estimation and updating $\widehat{g}_t$. This enables quicker adaptation to the new distribution. Note that the window size presents a trade-off: a smaller window encourages faster adaptation but results in a wider confidence interval, which in turn limits optimality.

9.5.2 Results and Discussions

C1. In stationary settings, ASAT aims to control FPR while maximizing TPR. In Figure 9.4, FSFT exhibits very high FPR at TPR-95%, whereas both FSAT and ASAT keep FPR consistently below 5% across all four settings. Initially, both FSAT and ASAT yield a TPR of 0% because insufficient OOD samples force $\widehat{\lambda}_t$ to be set to ∞ (as per Q1), until enough OOD samples are collected to satisfy $\psi_t(\delta) \leq \alpha$. Moreover, ASAT initially achieves a lower TPR than FSAT because it adapts both thresholds and scoring functions—resulting in a wider confidence bound $\psi_t(\lambda)$ that requires more OOD samples—while FSAT updates only thresholds (see Section 9.4).

However, this early drawback is offset by its ability to maximize TPR later in deployment. ASAT quickly catches up with FSAT and eventually surpasses the best achievable TPR by FSAT at FPR-5% in all settings. In particular, when both methods start with EBO, ASAT shows over a 40% improvement in TPR towards the end (see Figures 9.4(a) and 9.4(b)). As expected, $\text{TPR}(\widehat{g}_t, \widehat{\lambda}_t)$ converges to $\text{TPR}(\widehat{g}^\star, \lambda)$, confirming optimality. The stationary results verify that ASAT effectively updates both scoring functions and thresholds to maximize TPR while ensuring FPR control, outperforming the baselines.

C2. In nonstationary settings, ASAT aims to maximize TPR while minimizing FPR violation if it occurs. As shown in Figure 9.5, FPR violations occur when the new OOD is less distinguishable from the ID than the old OOD (e.g., Far-OOD to Near-OOD in Figures 9.5(b) and 9.5(d)) whereas no FPR violations are observed in the reverse shift (Figures 9.5(a) and 9.5(c)). In Figure 9.6, ASAT initially shows larger FPR violations than FSAT, which is expected since it adapts (or *specializes*) the scoring functions based on previous OOD data. However, ASAT quickly reduces FPR below α and effectively adapts to the new OOD, consistently achieving higher TPR than FSAT both before and after the shift.

Finally, Figure 9.7 illustrates that smaller windows enable faster adaptation and reduce FPR violations, although they limit the convergence of TPR.

9.6 Related Works

OOD Detection. Robustness to out-of-distribution (OOD) data has been extensively studied in recent years (Salehi et al., 2022; Yang et al., 2022b). Various post-hoc and training-time methods have been proposed to learn scoring functions for OOD detection (Bendale and Boult, 2015; Hendrycks and Gimpel, 2017b; Liu et al., 2020; Wang et al., 2022c; Sehwag et al., 2021; Ming et al., 2022; Ren et al., 2019; Kirichenko et al., 2020; DeVries and Taylor, 2018; Chen et al., 2021; Huang and Li, 2021; Wei et al., 2022c; Katz-Samuels et al., 2022b). However, most approaches rely solely on ID data to design scoring functions and set score thresholds, focusing on maximizing TPR. This approach often results in a high FPR. Furthermore, fixed scoring functions and thresholds limit a system's ability to adapt to varying OOD inputs.

FPR Control for OOD Detection. Some of these challenges were addressed by Vishwakarma et al. (2024c), who introduced a method that adapts thresholds over time—while using a fixed scoring function—to ensure FPR control. Although effective in controlling FPR, a fixed scoring function restricts the achievable TPR. Our work extends this approach by jointly learning scoring functions and thresholds to explicitly maximize TPR subject to an FPR constraint, using OOD inputs collected via human feedback during deployment. Similar ideas have been explored in model-assisted data labeling (Vishwakarma et al., 2024a) and in optimizing set sizes for conformal prediction (Stutz et al., 2022).

Time-uniform Confidence Sequences. Our method leverages time-uniform confidence sequences based on the Law-of-Iterated Logarithms (LIL) from multi-armed bandits literature (Darling and Robbins, 1967; Lai, 1976; Jamieson et al., 2014; Howard and Ramdas, 2022), which have been used in active learning for ranking and clustering settings (Heckel et al., 2019; Vinayak, 2018; Chen et al., 2023a). However, these anytime confidence intervals are often constructed under i.i.d. assumptions. Due to importance sampling with time-varying scoring functions and thresholds, the samples are no longer i.i.d. in ASAT. We address this by constructing a martingale sequence and building on the martingale-LIL result by Balsubramani (2015) to derive time-uniform confidence sequences suitable for our setting.

9.7 Conclusion

We introduced a human-in-the-loop OOD detection framework that updates both scoring functions and thresholds to maximize TPR while strictly controlling FPR. Our approach formulated this as a constrained joint optimization problem, with a relaxed version developed for practical learning. To guarantee FPR control throughout deployment, we designed novel time-uniform confidence sequences that account for dependencies from importance sampling and time-varying scores and thresholds. Empirical evaluations demonstrate that our approach consistently outperforms methods that use fixed scoring functions by maintaining FPR control and improving TPR.

9.8 Limitations and Future Work

While ASAT provides a new adaptive OOD detection framework with formal FPR control guarantees, several limitations remain and motivate future work.

Noisy Human Feedback. ASAT currently assumes accurate human labels but can potentially be extended to handle noisy feedback using techniques from the label noise literature, such as the unbiased estimators proposed by Natarajan et al. (2013b). Incorporating such corrections into FPR estimation and extending FPR guarantees under noisy labels is a promising direction for future work.

Shift Detection for Nonstationary Settings. Our guarantees assume stationary OOD distributions. A possible extension is to use change-point detection (e.g., by monitoring FPR) to detect shifts and re-initialize ASAT, which allows the stationary guarantees to apply piecewise across each stationary interval. A full theoretical analysis under nonstationary settings is left for future work.

Different Types of Distribution Shifts. ASAT focuses on semantic OOD shifts, but covariate shifts (e.g., noise, resolution changes) are also important. Evaluating and extending ASAT under such shifts is an open area for future research.

Chapter 10

Conformal Prediction for Language Models

Large language models (LLMs) are empowering decision-making in several applications, including tool or API usage and answering multiple-choice questions (MCQs). However, incorrect outputs pose significant risks in high-stakes domains like healthcare and finance. To quantify LLM uncertainty and thereby mitigate these risks, recent works employ conformal prediction (CP), a model- and distribution-agnostic framework that uses LLM outputs to generate a *prediction set* containing the true answer with high probability. Leveraging CP, we propose *conformal revision of questions* (CROQ), which revises the question by narrowing down the available choices to those in the prediction set and asking the LLM the revised question. We expect LLMs to be more accurate on revised questions with fewer choices. Furthermore, we expect CROQ to be effective when the prediction sets from CP are small. Commonly used logit scores often lead to large sets, diminishing CROQ's effectiveness. To overcome this, we propose CP-OPT, an optimization framework to learn scores that minimize set sizes while maintaining coverage. Our extensive experiments on MMLU, ToolAlpaca, and TruthfulQA datasets with multiple LLMs show that CROQ improves accuracy over the standard inference, with more pronounced gains when paired with CP-OPT.

10.1 Introduction

Large language models (LLMs) (Touvron et al., 2023; Databricks, 2024; Abdin et al., 2024) have demonstrated remarkable capabilities in various decision-making tasks, including multi-choice question answering and tool usage, where the model must select the correct tool or API to complete a task (Qu et al., 2024; Tang et al., 2023; Hendrycks et al., 2021). However, LLMs often exhibit overconfidence in wrong answers (Krause et al., 2023; Groot and Valdenegro Toro, 2024). Such unreliable predictions entail significant risks in critical domains like finance. Successful usage in such settings demands principled solutions to improve accuracy and quantify uncertainty in the predictions.

A commonly taught strategy for a human test taker to solve multi-choice questions (MCQs) is the process of elimination (pruning) of incorrect (distractor) answer choices. The underlying principle is that this enables them to focus their attention on the remaining answer choices, and it increases the likelihood of a correct answer even if they have to guess randomly. Inspired by this, we investigate whether LLMs can benefit from a similar strategy.

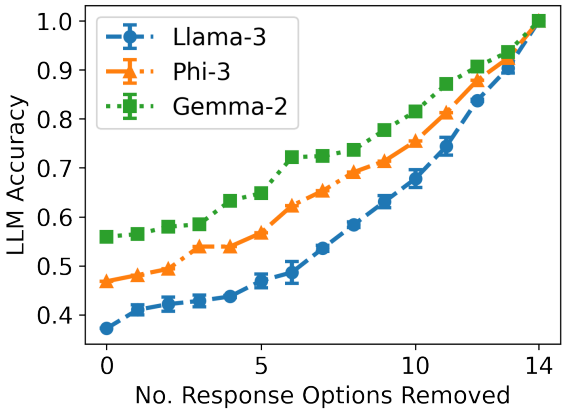


Figure 10.1: Accuracy for three LLMs on the TruthfulQA dataset with 15 response options as a function of the number of incorrect answer options (distractors) removed from the prompt. As more distractor answers are eliminated, accuracy increases. Accuracy is averaged across 5 iterations, error bars denote ±2 standard deviations.

We first examine the relationship between the number of distractor answers and LLM accuracy on a MCQ task. Figure 10.1 illustrates accuracy for three different LLMs on a version of TruthfulQA, a widely used MCQ dataset. The MCQs in this version of TruthfulQA have 15 answer options, only one of which is correct. (We discuss how this dataset is constructed in Appendix G.4.2.) For each question, we repeatedly prompt the LLM, randomly eliminating one distractor answer at a time. Each prompt is independent, without any previous rounds included

in the context. As hypothesized, *reducing the number of response options leads to an improvement in accuracy*, and this improvement is very nearly monotone. This suggests that eliminating distractor answers before prompting the LLM can indeed enhance accuracy. Of course, when pruning answers, we do not want to eliminate the correct answer, since that would necessarily cause the LLM to get the MCQ wrong.

Conformal prediction (CP) (Vovk et al., 2005b) is a flexible framework that can be used to prune distractor answers while retaining the correct answer with high probability. CP is a model-agnostic and distribution-free technique for generating prediction sets which contain the correct answer with a user-specified probability (e.g. 95%), which is referred to as the *coverage guarantee*.

Utilizing this guarantee of CP, we propose a procedure called *conformal revision of questions* (CROQ), to revise MCQs with choices in a prediction set output by CP. This procedure represents a tradeoff: with some small probability (e.g. 5%), we may remove the correct answer from the prediction set, causing the LLM to get the question wrong. However, with high probability (e.g. 95%), we will retain the correct answer while reducing the number of distractor answers. Given the relationship observed in Figure 10.1, this should increase the LLM's accuracy on those questions. Different coverage rates naturally induce different tradeoffs. Overall, we hypothesize that we can find a coverage rate with a favorable tradeoff, such that CROQ improves the overall accuracy on a given MCQ task.

CROQ's effectiveness depends on the size of the prediction sets from conformal prediction – smaller sets mean fewer choices in the revised question and hence better final accuracy. Conformal prediction requires specifying a *score function*, which loosely speaking quantifies how plausible an output (answer option) is for a given input (question). While conformal prediction provides a coverage guarantee for *any* score function, the size of the prediction sets depends on the score function. As an example, a random score function will yield output sets that constitute random subsets of the label space that are large enough to satisfy the coverage guarantee (Angelopoulos and Bates, 2022).

Previous works that apply conformal prediction in MCQ-type settings have used readily available

scores such as the logits (or softmax values) output from the LLM (Kumar et al., 2023) or have designed heuristic scores based for example on repeated querying of the LLM (Su et al., 2024). Logits can be overconfident and may show biases for some options (Zheng et al., 2024), and heuristic scores are not guaranteed to produce small sets. Thus, in order to make CROQ as effective as possible, we propose CP-OPT (conformal prediction optimization), a principled solution to obtain scores that are designed to minimize set sizes (uncertainty) while preserving the coverage guarantee.

To summarize, our main contributions are as follows:

1. We propose the conformal revision of questions (CROQ), in which we prune the answer choices in an MCQ to those in the prediction set output by conformal prediction and then prompt the LLM with the revised question. Empirical evaluation shows that this approach consistently improves accuracy compared to prompting the LLM with the original MCQ.
2. We design a score function optimization framework (CP-OPT) that can be applied to any pre-trained LLM. Moving away from the potentially unreliable LLM logits and heuristic scores, our framework provides a principled way to learn scores for conformal prediction. Empirically, we show that our procedure leads to a reduction in average set sizes compared to the baseline procedure that uses the LLM logits as the scores, at the same level (95%) of coverage.
3. We further show that when used with CROQ, our CP-OPT scores deliver greater accuracy improvements over baseline than the LLM's logits.

10.2 Preliminaries

In this section, we provide background on solving MCQ tasks with LLMs and conformal prediction.

10.2.1 Multiple Choice Questions (MCQs) and LLMs

MCQ Setup. MCQs are a general abstraction for expressing problems in which the correct choice(s) must be selected from a given set of choices. These encompass question-answering tasks

like MMLU (Hendrycks et al., 2021) as well as other tasks such as tool learning, in which the LLM must select the correct tool or API to complete a task (Tang et al., 2023; Qu et al., 2024). An MCQ consists of the question text Q, i.e. a sequence of tokens, and a set of answer choices $O = \{(Y_1, V_1), (Y_2, V_2), \ldots, (Y_m, V_m)\}$. Here, each Y_j is a unique character from the English alphabet, and we assume that the number of choices m is less than or equal to the size of the alphabet. Each V_j is the option text for the jth option. Denote the whole MCQ instance as $x = (Q, O)$. Let $\mathcal{X}_m$ denote the space of MCQs with m choices and $\mathbb{P}_{\mathcal{X}_m}$ denote a distribution over $\mathcal{X}_m$, from which samples for training, calibration, and testing are drawn independently. Here, we assume that for each question Q there is only one correct answer key $y^\star \in \{Y_1, Y_2, \ldots Y_m\} = \mathcal{Y}_m$.

MCQ Prompt. We concatenate the question text Q and the answer choices O, all separated by a new line character, and append to the end the text "`The correct answer is: `". The expectation is that given this input prompt, the next token predicted by the LLM will be one of the option keys. See Appendix G.4 for a prompt example. We consider zero-shot prompts and do not include example questions and answers in the prompt. We also add the prefix and suffix tokens to the prompt as recommended by the language model providers. Since these are fixed modifications to x, we will use x to denote the final prompt and the MCQ instance analogously.

LLM Inference. We run the forward pass of the auto-regressive LLM (Touvron et al., 2023; Dubey et al., 2024; Abdin et al., 2024) on the input prompt to obtain the logit scores for each possible next token given the prompt, restricting attention to the tokens that correspond to the available answer keys (e.g. "a", "b", "c", "d" if there are four answer options). We take the softmax to convert the logits to probabilities, and then we take as the LLM's answer the option with the highest probability. This approach ensures that the LLM's answer will be one of the available answer options, which would not be guaranteed if instead we asked the LLM to simply generate an answer token given the prompt. This approach mirrors what has been done in other works that use LLMs to solve MCQs (Kumar et al., 2023; Su et al., 2024). Formal details are given in Appendix G.1.1.

10.2.2 Conformal Prediction

Conformal prediction (CP) (Vovk et al., 2005b; Angelopoulos et al., 2022) is a framework for quantifying uncertainty in machine learning models. It provides a flexible and user-friendly approach to output *prediction sets* (which may be finite sets or intervals) that contain the true output or label with a probability that is specified by the user, e.g. 95%. The key strength of conformal prediction lies in its *distribution-free* guarantees: it ensures that the constructed prediction sets are valid regardless of the underlying data distribution and model. This property is particularly desirable in the context of language models, as it is hard to characterize language data distributions or put specific distributional assumptions/restrictions on the LLMs.

Score Function. Let $g : \mathcal{X}_m \times \mathcal{Y}_m \mapsto \mathbb{R}$ be a conformal *score function*, where larger scores indicate better agreement ("conformity") between x and y. Intuitively, large scores are intended to indicate that y is a plausible output given x, while smaller scores indicate less plausibility. (Note that some authors prefer to have larger scores indicate greater disagreement, e.g. Clarkson et al. (2024).) A common choice of score function is the softmax scores from the given model. For closed-source LLMs, where logits are not available, others have devised self-consistency scores based on repeated querying of the model (Su et al., 2024).

Prediction Sets. Given a score function g and threshold τ on the scores, the prediction set for any $x \in \mathcal{X}_m$ is given by

$$C(x; g, \tau) := \{y \in \mathcal{Y}_m : g(x, y) \geq \tau\}. \tag{10.1}$$

Intuitively, larger sets represent greater uncertainty, while smaller sets represent less uncertainty. Given a fixed confidence level, a score function that produces larger sets can be said to result in greater uncertainty.

Split Conformal Prediction. Similar to prior works (Kumar et al., 2023; Su et al., 2024), we use *Split Conformal Prediction* (Papadopoulos et al., 2002; Lei et al., 2018) due to its popularity, ease of use, and computational efficiency. Given a score function $g : \mathcal{X}_m \times \mathcal{Y}_m \mapsto \mathbb{R}$, Split Conformal Prediction uses a calibration dataset $D_{\text{cal}} = \{x_i, y_i^\star\}_{i=1}^{n_{\text{cal}}}$ to compute a threshold $\hat{\tau}_\alpha$,

defined as

$$\hat{\tau}_\alpha = \min \left\{ q : \frac{1}{n_{\text{cal}}} \sum_{i=1}^{n_{\text{cal}}} \mathbb{1}\left(g(x_i, y_i^\star) \leq q\right) \geq \frac{\lfloor (n_{\text{cal}} + 1)\alpha \rfloor}{n_{\text{cal}}} \right\}, \tag{10.2}$$

where $\alpha \in [0, 1]$ is a user-chosen *miscoverage rate* that is equal to 1 minus the desired coverage; for example, a value of $\alpha = 0.05$ would correspond to a coverage of 95%. In words, $\hat{\tau}_\alpha$ is the smallest empirical quantile of the scores for the correct answers on the calibration dataset that is sufficient to satisfy (an empirical version of) the coverage property. The threshold $\hat{\tau}$ is used to construct prediction sets $C(x; g, \hat{\tau})$ on previously unseen test points as in (10.1). The "split" terminology refers to the fact that the data which is used to compute $\hat{\tau}$ is separate from the data for which prediction sets are constructed. This procedure enjoys a marginal coverage guarantee for prediction sets on unseen test data points, formalized as Proposition 2.

Proposition 2. (Marginal Coverage Guarantee) (Lei et al., 2018, Thm. 2.2) Let g be a fixed conformity score function and $\hat{\tau}_\alpha$ be an α threshold computed via Split Conformal Prediction on $D_{\text{cal}} = \{x_i, y_i^\star\}_{i=1}^{n_{\text{cal}}} \sim \mathbb{P}_{\mathcal{X}_m \times \mathcal{Y}_m}$. Then, for a new sample $(\tilde{x}, \tilde{y}^\star) \sim \mathbb{P}_{\mathcal{X}_m \times \mathcal{Y}_m}$, we have that

$$\mathbb{P}(\tilde{y}^\star \in C(\tilde{x}\,;\, g, \hat{\tau}_\alpha)) \geq 1 - \alpha. \tag{10.3}$$

where the probability is marginal over the randomness in the calibration data and the new sample.

The top half of Figure 10.2 illustrates conformal prediction for answering MCQs with LLMs. While the coverage guarantee in Proposition 2 holds for any score function, ideally we would like a score function that yields the smallest sets possible (the least uncertainty). Next, we discuss our solutions to improve conformal prediction and its utility in solving MCQs with LLMs.

10.3 Methodology

In this section, we discuss details of our pipeline for question revision using conformal prediction and our procedure to generate optimal conformal scores.

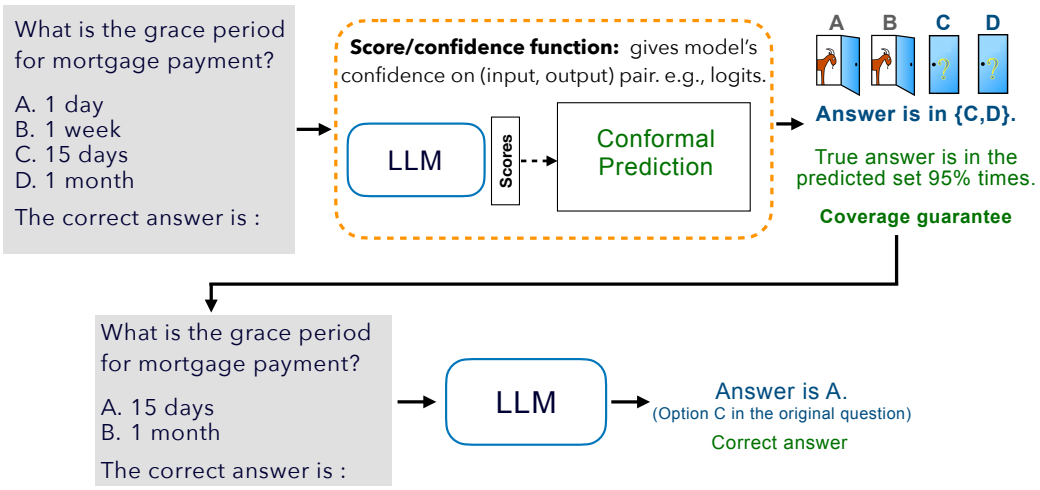


Figure 10.2: (CROQ) Illustration of conformal revision of questions and prompting the LLM with the revised question. In this example, the initial predicted set by LLM + conformal prediction (CP) is {C, D}. The question and labels are revised to contain only the answer choices in the prediction set and the LLM is prompted with the revised question. Since CP provides rigorous coverage guarantees, we expect that re-prompting the LLM with reduced answer choices will improve the chances of obtaining the correct answer. See Section 10.3.1 for more details.

10.3.1 Conformal Revision of Questions (CROQ)

The procedure involves prompting the LLM with the reduced answer options from a conformal prediction set. The steps are illustrated with an example in Figure 10.2.

Scores and Threshold for Conformal Prediction. We first fix a score function $g : \mathcal{X}_m \times \mathcal{Y}_m \mapsto \mathbb{R}$. Here we restrict the score function to either the logits generated by the LLM or the CP-OPT scores discussed in Section 10.3.2. We then run the split conformal procedure with coverage level $1-\alpha$ for some $\alpha \in [0,1]$ to estimate the threshold $\hat{\tau}_\alpha$. CROQ then proceeds as follows.

Step 1: Get Conformal Prediction Set. Given a test instance x, we generate a first stage prediction set, $C(x\,; g, \hat{\tau}_\alpha)$. Per the coverage guarantee (Proposition 2), we expect that the true answer $y^\star \in C(x\,; g, \hat{\tau}_\alpha)$ with probability at least $1-\alpha$. Next, the question is revised to contain only the choices in the set $C(x\,; g, \hat{\tau}_\alpha)$.

Step 2: Revise the Question and Ask the LLM. If the first stage prediction set $C(x\,; g, \hat{\tau}_\alpha)$ is empty or is of size 1 or size m (the number of answer options), then we simply utilize the LLM's

answer to the original MCQ x, as described in section 10.2.1, since the conformal procedure has yielded no additional information. Otherwise, we modify the prompt x to $x' = (Q, O')$, where $O' = \{(K_j, V_j) : K_j \in C(x\,; g, \hat{\tau}_\alpha)\}$. The keys in O' are changed so that they start with the first letter of the alphabet and go to the letter corresponding to the number of choices available. For example, if there were initially four answer options {a, b, c, d}, and the conformal prediction set was {c, d}, then the two options in the set would receive new keys {a, b}. Then x' is transformed into a prompt format and passed to the LLM, and the standard inference procedure (section 10.2.1) is run to extract the predicted answer key $\hat{y}'$.

With fewer choices in the revised question, we expect LLMs will be more accurate in their answer compared to the answer to the initial question. However, the improvement in accuracy will depend on the size of the prediction sets. As illustrated in Figure 10.1 LLMs have a higher chance of answering the question correctly if the number of response options is small. This implies the efficacy of CROQ will depend on the size of sets $C(x; g, \hat{\tau}_\alpha)$. If these sets are small then we can expect more improvement from CROQ.

While conformal prediction can output sets $C(x; g, \hat{\tau}_\alpha)$ for any score function g, along with $1 - \alpha$ coverage guarantee, the set sizes could be highly variable depending on the score function g. Noting the lack of reliability of scores used in prior works, that could yield unnecessarily large sets, we seek to learn scores that minimize the set sizes while preserving the coverage guarantee. We discuss our procedure to learn such scores in the next section. Using these scores in CP we expect to get smaller sets and thus more improvement in CROQ compared to baseline scores.

10.3.2 CP-OPT to Optimize Scores

We describe our method for learning the optimal scores for conformal prediction (CP) for solving MCQs with LLMs. Similar ideas have been incorporated in the training objective of classifiers (Stutz et al., 2022) so that the classifiers' softmax output is better suited for CP. However, the LLMs are not trained with this objective, and we want to apply CP to any given LLM; therefore, we design a post-hoc method to optimize the scores. We first characterize the optimal scores and then describe how to estimate them in practice.

Characterization of the optimal scores. For any score function $g : \mathcal{X}_m \times \mathcal{Y}_m \mapsto \mathbb{R}$ and threshold τ, the membership of any y in the prediction set $C(x \mid g, \tau)$ is given by $\mathbb{1}(y \in C(x \mid g, \tau)) = \mathbb{1}\{g(x, y) \geq \tau\}$. Define the expected set size $S(g, \tau)$ and the coverage conditional on τ, denoted $\mathscr{P}(g, \tau)$, as follows:

$$S(g, \tau) := \mathbb{E}_x\Big[\sum_{y \in \mathcal{Y}_m} \mathbb{1}\{g(x, y) \geq \tau\}\Big]. \tag{10.4}$$

$$\mathscr{P}(g, \tau) := \mathbb{E}_x\left[\mathbb{1}\{g(x, y^\star) \geq \tau\}\right]. \tag{10.5}$$

The optimal score function $g^\star$ and threshold $\tau^\star$ are defined (non-uniquely) to minimize the expected set size subject to the coverage $\mathscr{P}(g, \tau)$ being at least $1 - \alpha$:

$$g^\star, \tau^\star := \underset{g: \mathcal{X}_m \times \mathcal{Y}_m \mapsto \mathbb{R}, \tau \in \mathbb{R}}{\arg\min} \; S(g, \tau) \;\text{ s.t. }\; \mathscr{P}(g, \tau) \geq 1 - \alpha. \tag{P10.1}$$

Practical Version with Differentiable Surrogates and Empirical Estimates. Problem (P10.1) characterizes optimal score functions and thresholds. However, in practice, we do not know the underlying distribution and thus do not have access to the quantities in (10.4) and (10.5). Instead, we obtain their estimates using a training sample $D_{\text{train}} = \{(x_i, y_i^\star)\}_{i=1}^{n_t}$ drawn independently from the same distribution:

$$\widehat{S}(g, \tau) := \frac{1}{n_t} \sum_{i=1}^{n_t} \sum_{y \in \mathcal{Y}_m} \mathbb{1}\{g(x_i, y) \geq \tau\}, \tag{10.6}$$

$$\widehat{\mathscr{P}}(g, \tau) := \frac{1}{n_t} \sum_{i=1}^{n_t} \mathbb{1}\{g(x_i, y_i^\star) \geq \tau\}. \tag{10.7}$$

Using these plug-in estimators in problem (P10.1) yields a revised optimization problem. However, it is difficult to solve this problem as the objective and constraints are not differentiable. To make them differentiable, we introduce the following surrogates. Given $g(x, y)$ and τ, define the following sigmoid function with $\beta > 0$, $\sigma(x, y, g, \tau, \beta) := 1/\big(1 + \exp(-\beta\,(g(x, y) - \tau))\big)$. The

sigmoid function provides a differentiable approximation to the indicator variable for $g(x, y) \geq \tau$. The approximation is tighter with larger β i.e., $\sigma(x, y, g, \tau, \beta) \to \mathbb{1}\{g(x, y) \geq \tau\}$ as $\beta \to \infty$, and $g(x, y) \geq \tau \iff \sigma(x, y, g, \tau) \geq 1/2$. By using these sigmoid surrogates in equation (10.6), we obtain the following smooth plugin estimates,

$$\widetilde{S}(g, \tau) := \frac{1}{n_t} \sum_{i=1}^{n_t} \sum_{y \in \mathcal{Y}_m} \sigma(x_i, y, g, \tau, \beta). \tag{10.8}$$

$$\widetilde{\mathscr{P}}(g, \tau) := \frac{1}{n_t} \sum_{i=1}^{n_t} \sigma(x_i, y_i^\star, g, \tau, \beta). \tag{10.9}$$

It is easy to see that by the strong law of larger numbers and properties of the sigmoid function, as $n_t, \beta \to \infty$, the surrogate average set size and coverage will converge almost surely to their population versions, i.e. $\widetilde{S}(g, \tau) \xrightarrow{a.s.} S(g, \tau)$ and $\widetilde{\mathscr{P}}(g, \tau) \xrightarrow{a.s.} \mathscr{P}(g, \tau)$. We replace the expected set size and marginal coverage by these smooth surrogates in (P10.1) and transform it into an unconstrained problem with a penalty term $\lambda > 0$. We also introduce ℓ_2 regularization to encourage low norm solutions. We optimize the score function g over a flexible space of functions $\mathcal{G}$, such as neural networks (NNs). The resulting problem (P10.2) is differentiable, and we solve it using stochastic gradient descent.

$$\tilde{g}, \tilde{\tau} := \underset{g \in \mathcal{G}, \tau \in \mathbb{R}}{\arg\min}\, \widetilde{S}(g, \tau) + \lambda\big(\widetilde{\mathscr{P}}(g, \tau) - 1 + \alpha\big)^2 - \hat{\mathcal{C}}(g) + \lambda_1 \|g\|_2^2. \tag{P10.2}$$

Here, $\hat{\mathcal{C}}(g) := \frac{1}{n_t} \sum_{i=1}^{n_t} \log(g(x_i, y_i^*))$ is the cross entropy term included to encourage higher scores for correct predictions, and the regularization term $\lambda_1 ||g||_2^2$ is the squared norm over the parameters of g to promote low norm solutions. Solving (P10.2) yields a score function $\tilde{g}$ and a threshold $\tilde{\tau}$. However, $\tilde{\tau}$ may be biased, since it is estimated on the same data as $\tilde{g}$. Following the split conformal procedure, we therefore estimate a new threshold $\hat{\tau}$ on a separate calibration dataset. Note that our framework is flexible and can work with any choice of features and function class for which the ℓ_2 norm can be calculated. We discuss the specific choice of features and $\mathcal{G}$

used in this work.

Specific choice of features and $\mathcal{G}$. In practice, we want to use a flexible and easy-to-train function class for $\mathcal{G}$. As this is a post-hoc procedure and we want to avoid expensive fine-tuning. We use 3-layer neural networks with `tanh` activation as $\mathcal{G}$ and use the LLM's logits and the penultimate layer's representations corresponding to the last token as input features to the g network. Let $\boldsymbol{z} \in \mathbb{R}^{d+m}$ be the concatenation of the LLM's penultimate layer's representation (d-dimensional) and logits (m-dimensional) for the last token. Our choice of $\mathcal{G}$ for the experiments is defined as follows,

$$\mathcal{G} := \{\, g : \mathbb{R}^{d_0} \to \Delta^{m-1} \mid g(\boldsymbol{z}) := \texttt{softmax}(\boldsymbol{W}_3\texttt{tanh}(\boldsymbol{W}_2\texttt{tanh}(\boldsymbol{W}_1(\boldsymbol{z})))), \\ \boldsymbol{W}_1 \in \mathbb{R}^{d_0 \times d_1}, \boldsymbol{W}_2 \in \mathbb{R}^{d_1 \times d_2}, \boldsymbol{W}_3 \in \mathbb{R}^{d_2 \times m} \,\}$$

Here, $d_0 = d + m, d_1 = (d + m)/2$, and $d_3 = (d + m)/4$ and Δ^{m-1} is the $m - 1$ dimensional probability simplex. This class for $\mathcal{G}$ is flexible enough and the resulting optimization problem is not computationally prohibitive to solve. More complex (flexible) choices of $\mathcal{G}$ could be used when we can devote more compute to learning the score function.

10.4 Experiments

We conduct experiments on benchmark MCQ and tool usage tasks with open-weight instruction-tuned models to test the following hypotheses:

H1. CP-OPT scores in conformal prediction on MCQ tasks with LLMs yield a smaller average set size at the same level of coverage in comparison to using LLM logits.

H2. Conformal revision of questions (CROQ) improves accuracy over the standard inference procedure.

H3. CROQ with CP-OPT scores performs better than CROQ with logit scores.

		Llama-3				Phi-3				Gemma-2			
		Avg. Set Size		Coverage		Avg. Set Size		Coverage		Avg. Set Size		Coverage	
Dataset	# Opt.	Logits	Ours	Logits	Ours	Logits	Ours	Logits	Ours	Logits	Ours	Logits	Ours
MMLU	4	2.56	**2.53***	95.75	95.57	2.21	**2.16***	94.65	94.35	2.94	**2.40***	95.16*	94.23
	10	5.53	**4.90***	96.06*	95.45	4.36	**4.36**	94.11	94.09	7.79	**6.08***	95.00*	94.04
	15	7.69	**7.18***	95.42	95.06	6.64	**6.52***	94.60	94.61	11.71	**10.04***	94.58	94.58
ToolAlpaca	4	**1.17**	1.18	97.08	96.85	**1.07**	1.08	95.33	95.68	1.12	**1.05***	95.68	95.44
	10	1.51	**1.39***	95.21	95.56	1.25	**1.20***	95.56	95.09	2.05	**1.42***	95.56	94.51
	15	1.97	**1.67***	96.50	96.03	1.68	**1.54***	98.36*	97.20	3.54	**1.77***	96.14	95.21
TruthfulQA	4	3.34	**2.69***	95.95*	92.41	2.85	**2.53***	96.71	96.71	2.74	**1.88***	96.46	95.44
	10	7.06	**6.41***	94.43	93.42	7.48	**6.49***	98.48*	95.70	7.52	**5.64***	95.44	97.22
	15	**10.61**	10.62	94.68	94.68	10.72	**10.30***	95.44	96.46	11.23	**9.35***	95.44	96.46

Table 10.1: Average set sizes and coverage rates (in percentages) for conformal prediction sets on the MMLU, ToolAlpaca, and TruthfulQA datasets using `gemma-2-9b-it-SimPO` (Gemma-2), `Llama-3-8B-Instruct` (Llama-3) and `Phi-3-4k-mini-Instruct` (Phi-3), with a target coverage level of 95%. Bold numbers indicate smaller average set sizes. Asterisks on the larger of a pair of numbers indicate where the difference in average set size or coverage is statistically significant at the 0.05 significance level.

10.4.1 Experimental Setup

We first describe the setup for the experiments and then discuss the results for the above hypotheses.

Datasets. We evaluate our hypotheses on 3 datasets: `MMLU` (Hendrycks et al., 2021), `TruthfulQA` (Lin et al., 2022), and `ToolAlpaca` (Tang et al., 2023). MMLU and TruthfulQA are popular benchmark datasets for multiple-choice questions. MMLU focuses on assessing multitask accuracy; it contains multiple choice questions (MCQs) from 57 domains, including humanities, math, medicine, etc. TruthfulQA evaluates an LLM's ability to answer truthfully and avoid falsehoods that humans are susceptible to. ToolAlpaca contains 3.9k tool-use instances from a multi-agent simulation environment, which we augment to a MCQ format. Dataset descriptions and example questions and responses are provided in Appendix G.4.

Models. We use auto-regressive language models based on the transformer architecture. We choose instruction-tuned, open-weight, and small to medium-sized models, for reproducibility and reduced computational cost. Specifically, we use `Llama-3-8B-Instruct` by Meta (Dubey et al., 2024), `Phi-3-4k-mini-Instruct` by Microsoft (Abdin et al., 2024), and the `gemma-2-9b-it-SimPO`

Model	# Opt.	Llama-3			Phi-3			Gemma-2		
		Accuracy Before (a_1)	**Accuracy After** (a'_1)	**Gain** ($a'_1 - a_1$)	**Accuracy Before** (a_1)	**Accuracy After** (a'_1)	**Gain** ($a'_1 - a_1$)	**Accuracy Before** (a_1)	**Accuracy After** (a'_1)	**Gain** ($a'_1 - a_1$)
MMLU	4	**64.02**	63.83	-0.19	**70.27**	69.08	-1.19	67.62	**67.70**	**0.07**
	10	54.82	**56.29**	**1.47***	58.44	**61.57**	**3.13***	53.80	**53.93**	**0.13**
	15	51.99	**54.11**	**2.11***	53.48	**58.09**	**4.62***	**50.78**	50.58	-0.20
ToolAlpaca	4	91.47	**91.94**	**0.47**	**92.76**	92.64	-0.12	**93.46**	93.11	-0.35
	10	85.16	**88.67**	**3.50***	87.50	**90.89**	**3.39***	87.73	**89.60**	**1.87***
	15	81.43	**87.85**	**6.43***	85.98	**89.25**	**3.27***	87.97	**88.55**	**0.58**
TruthfulQA	4	54.43	**55.19**	**0.76**	69.87	**70.13**	**0.25**	74.68	**74.94**	**0.25**
	10	39.24	**40.76**	**1.52**	**55.70**	54.43	-1.27	**56.46**	56.20	-0.25
	15	37.22	**37.22**	**0.00**	**46.84**	46.33	-0.51	55.95	**56.96**	**1.01**

Table 10.2: [CROQ + logits]. Results on accuracy improvement with CROQ using logit scores. Here a_1, and a'_1 refer to the accuracy before CROQ and after CROQ respectively. A positive gain implies CROQ improved accuracy in that setting.

Model	# Opt.	Llama-3			Phi-3			Gemma-2		
		Accuracy Logits (a'_1)	**Accuracy CP-OPT** (a'_2)	**Gain** ($a'_2 - a'_1$)	**Accuracy Logits** (a'_1)	**Accuracy CP-OPT** (a'_2)	**Gain** ($a'_2 - a'_1$)	**Accuracy Logits** (a'_1)	**Accuracy CP-OPT** (a'_2)	**Gain** ($a'_2 - a'_1$)
MMLU	4	**63.83**	63.67	-0.16	69.08	**69.34**	**0.26**	67.70	**69.56**	**1.86***
	10	56.29	**57.11**	**0.82***	**61.57**	61.05	-0.52	53.93	**57.93**	**4.00***
	15	54.11	**54.77**	**0.66***	58.09	**58.15**	**0.06***	50.58	**51.31**	**0.73**
ToolAlpaca	4	**91.94**	91.82	-0.12	**92.64**	92.52	-0.12	93.11	**93.57**	**0.46**
	10	88.67	**89.02**	**0.35***	90.89	**91.00**	**0.11***	89.60	**90.42**	**0.82***
	15	87.85	**88.67**	**0.82***	89.25	**89.95**	**0.70***	88.55	**89.37**	**0.82**
TruthfulQA	4	55.19	**55.44**	**0.25**	**70.13**	69.87	-0.26	74.94	**76.96**	**2.02**
	10	40.76	**42.28**	**1.52**	54.43	**56.20**	**1.77**	56.20	**60.76**	**4.56***
	15	37.22	**37.47**	**0.25**	46.33	**51.39**	**5.06***	56.96	**57.72**	**0.76**

Table 10.3: [CROQ + logits vs CROQ + CP-OPT]. Comparison of CP-OPT and logits on accuracy improvement with CROQ. Here, a'_1, and a'_2 are the final accuracies after CROQ using logits and CP-OPT respectively (as in Tables 10.2 and G.1. The gain $a'_2 - a'_1$ is the difference between these two, with values indicating more improvement in CROQ with CP-OPT scores.

model (Meng et al., 2024). For brevity, we use the short names Llama-3, Phi-3, and Gemma-2 respectively for these models.

Choices of Scores. We use the following scores for conformal prediction. `(1) LLM Logits (Softmax)` are extracted from the LLM as discussed in Section 10.2.1. These have been used in prior works (Kumar et al., 2023; Su et al., 2024). `(2) CP-OPT (Ours)` are the scores learned using the score optimization procedure discussed in Section 10.3.2. We use the train split for each dataset to learn these scores. The hyperparameter settings we used for CP-OPT are given in Appendix G.4.3. We omit the self-consistency based heuristic scores proposed by Su et al. (2024), as these require repeated inferences to get good estimates of the scores, and hence have a high computational cost.

We use the provided validation splits as our calibration datasets for the conformal procedure. For testing the hypotheses, we calibrate the conformal threshold for the coverage guarantee of 95%, i.e. we set the miscoverage rate α to 0.05. In addition, we study CROQ with calibration in a range of α values: {0.01, 0.02, 0.03, 0.04, 0.05, 0.06, 0.07, 0.08, 0.09, 0.1, 0.15, 0.2, 0.25, 0.3, 0.4, 0.5 }. Performance is computed on test splits. The hyperparameters used to learn the score function using SGD are provided in table G.17 in Appendix G.4.3.

Statistical Significance. We report the statistical significance of our results using paired sample t-tests, using asterisks (*) to annotate results that are statistically significant at a 0.05 significance level. See Appendix G.3 for details.

10.4.2 Discussion

H1. Improvement in conformal set sizes with our CP-OPT scores. We run the CP procedure using the LLM logits and CP-OPT scores and obtain conformal sets for points in the test sets. We compute the average set size and coverage for each dataset, model, and score combination. The results are in Table 10.1. As expected, in most settings (17 out of 27) we see a statistically significant reduction in the set sizes with our (CP-OPT) scores with similar coverage as logits. The reduction is more pronounced with a higher number of options. In a few settings

(6/27), the reduction in set size is accompanied by a statistically significant decrease in coverage relative to using the logits. In the remaining 4/27 settings the differences are insignificant. Note that since the target coverage level is 95%, anything above 95% is over-coverage. We see that logits tend to over-cover and thus a drop in coverage is expected as long as it does not fall significantly below the desired level of 95% (this happens only in 2/27 settings). Overall, these results show CP-OPT's effectiveness in reducing set sizes while maintaining the target coverage level. In Appendix G.2, we provide histograms (e.g., Figure G.4) of set sizes produced by logits and CP-OPT scores in all settings. These histograms show a clear pattern: CP-OPT scores produce fewer large sets and more small sets in comparison to logit scores.

H2. Accuracy improvement with conformal revision of questions (CROQ). Tables 10.2 and G.1 show the accuracy before and after CROQ with logit and CP-OPT scores respectively. With the logit scores (Table 10.2), we see an increase in accuracy (by up to 6.43%) in 19 out of 27 settings, out of which 9 are statistically significant. In 8 of the settings, we see a small drop in accuracy (which is not statistically significant). Next, with CP-OPT scores (Table G.1) we see accuracy improvements (up to 7.24%) in 24 settings, of which 13 are statistically significant. In the remaining 3 settings, we see a non-significant drop in accuracy. Overall, we observe that in the vast majority of the settings, CROQ improves accuracy with either logits or CP-OPT scores. The rare small drops in accuracy could occur since the conformal procedure may eliminate the correct option with low probability (α).

H3. CROQ with CP-OPT scores is better than CROQ with logit scores. CP-OPT scores are designed to minimize set sizes while maintaining the coverage guarantee. As a result, using these scores with CROQ is expected to reduce uncertainty for many questions, leading to fewer answer options in the revised prompts. Based on Figure 10.1, we expect LLMs to be more likely to answer correctly when prompted with the revised question with fewer options. The results of CROQ with CP-OPT are summarized in Table G.1, and in Table 10.3 we compare the accuracies after CROQ with logits and CP-OPT. In Table 10.3 we see that in 22 out of 27 settings, CROQ with CP-OPT results in higher accuracy (up to 4.56%) than CROQ with logits. Furthermore, the improvements in 12 out of these 22 settings are statistically significant. The

drop in accuracy in the remaining 5 settings is statistically non-significant. Overall, the results show that CROQ with CP-OPT is generally better than with logits.

10.5 Related Work

Conformal Prediction for Uncertainty Quantification with LLMs. Recently there has been growing interest in using conformal prediction to quantify and control uncertainty in LLM-related tasks. In the context of multi-choice question answering (MCQ), previous works have investigated a variety of conformal score functions, including (the softmax of) the LLM logits corresponding to the response options (Kumar et al., 2023; Ren et al., 2023b) or functions thereof (Ye et al., 2024), confidence scores generated by the LLM itself, or "self-consistency" scores derived by repeated querying of the LLM (Su et al., 2024). We build on this work by aiming to learn a conformal score function that yields small conformal sets, rather than taking the score function as given.

In addition to the MCQ setting, there has been recent work utilizing conformal prediction in the context of open-ended response generation (Quach et al., 2024; Mohri and Hashimoto, 2024; Cherian et al., 2024). This setting differs in that there is not necessarily a unique correct response, so the notion of coverage must be redefined around *acceptability* or *factuality* rather than correctness. When factuality is the target, the goal is to calibrate a pruning procedure that removes a minimal number of claims from an LLM-generated open response, such that the remaining claims are all factual with high probability; that is, the goal is to retain as large a set as possible, rather than to generate a set with the smallest number of responses possible as in MCQ. Conformal prediction has also been used to capture token-level uncertainty (Deutschmann et al., 2024; Ravfogel et al.; Ulmer et al., 2024).

Optimizing Conformal Prediction Procedures. Several recent works have considered how to learn good conformal score functions from data, primarily in the context of supervised learning models (Bai et al., 2022; Stutz et al., 2022; Yang and Kuchibhotla, 2024; Xie et al., 2024). With LLMs, Cherian et al. (2024) consider how to learn a good score function to achieve factuality

guarantees; their optimization problem differs from ours due to the difference in setting as well as the addition of conditional coverage constraints (ensuring that coverage holds in different parts of the feature space). Kiyani et al. (2024) design a framework to minimize the size ("length," in their terminology) of conformal sets, which they apply to MCQ as well as to supervised learning problems. However, their framework is concerned with how to generate sets given a model and a conformity score, rather than how to learn a conformity score.

The works mentioned above all aim to produce small conformal sets that satisfy coverage guarantees. Among these, only Ren et al. (2023b) consider how conformal sets may be used downstream, in their case to improve the efficiency and autonomy of robot behavior. To our knowledge, our work is the first to investigate whether conformal prediction can be used to increase the accuracy of LLMs on MCQ type tasks.

10.6 Conclusions and Future Works

In this work, we introduced Conformal Revision of Questions (CROQ), a principled approach to improve LLM accuracy in multiple-choice settings by leveraging conformal prediction (CP) to eliminate distractor answers while maintaining high coverage of the correct answer. To further boost CROQ's performance we proposed CP-OPT, a framework for optimizing score functions to minimize prediction set sizes while preserving CP's coverage guarantees. Our results demonstrate that CROQ significantly enhances LLM's accuracy, and that CP-OPT further strengthens this effect by producing smaller, more reliable prediction sets than standard LLM logits. These findings highlight the potential of uncertainty-aware, test-time methods to improve LLM decision-making, providing a principled path for safer and more effective deployment of LLMs in critical applications.

Future works could explore multi-round CROQ, where answer options are pruned iteratively in multiple rounds, further improving accuracy while maintaining coverage. This requires developing efficient recalibration strategies and methods to prevent excessive coverage reduction across iterations. Additionally, a key challenge is adapting conformal score thresholds in settings with

a variable number of response options. Techniques like quantile regression could help calibrate thresholds dynamically, ensuring robust performance across diverse decision-making scenarios.

Disclaimer

This chapter and Appendix G were prepared for informational purposes by the Artificial Intelligence Research group of JPMorgan Chase & Co. and its affiliates ("JP Morgan") and is not a product of the Research Department of JP Morgan. JP Morgan makes no representation and warranty whatsoever and disclaims all liability, for the completeness, accuracy or reliability of the information contained herein. This document is not intended as investment research or investment advice, or a recommendation, offer or solicitation for the purchase or sale of any security, financial instrument, financial product or service, or to be used in any way for evaluating the merits of participating in any transaction, and shall not constitute a solicitation under any jurisdiction or to any person, if such solicitation under such jurisdiction or to such person would be unlawful.

Bibliography

Changho Shin, Winfred Li, Harit Vishwakarma, Nicholas Carl Roberts, and Frederic Sala. Universalizing weak supervision. In *International Conference on Learning Representations*, 2022a.

Harit Vishwakarma, Heguang Lin, Frederic Sala, and Ramya Korlakai Vinayak. Promises and pitfalls of threshold-based auto-labeling. *Advances in Neural Information Processing Systems*, 36:51955–51990, 2023.

Harit Vishwakarma, Yi Chen, Sui Jiet Tay, Satya Sai Srinath Namburi, Frederic Sala, and Ramya Korlakai Vinayak. Pearls from pebbles: Improved confidence functions for auto-labeling. *Advances in Neural Information Processing Systems*, 37:15983–16015, 2024a.

Harit Vishwakarma, Yi Chen, Satya Sai Srinath Namburi GNVV, Sui Jiet Tay, Ramya Korlakai Vinayak, and Frederic Sala. Pablo: Improving semi-supervised learning with pseudolabeling optimization. In *NeurIPS 2024 Workshop: Self-Supervised Learning-Theory and Practice*, 2024b.

Harit Vishwakarma and Frederic Sala. Lifting weak supervision to structured prediction. *Advances in Neural Information Processing Systems*, 35:37563–37574, 2022.

Harit Vishwakarma, Heguang Lin, and Ramya Korlakai Vinayak. Taming false positives in out-of-distribution detection with human feedback. In *Proceedings of the 27th International Conference on Artificial Intelligence and Statistics (AISTATS)*, volume 238. PMLR, 2024c.

Daisuke Yamada, Harit Vishwakarma, and Ramya Korlakai Vinayak. Adaptive scoring and thresholding with human feedback for robust out-of-distribution detection. *arXiv preprint arXiv:2505.02299*, 2025.

Harit Vishwakarma, Alan Mishler, Thomas Cook, Niccolo Dalmasso, Natraj Raman, and Sumitra Ganesh. Monty hall and optimized conformal prediction to improve decision-making with llms. *arXiv preprint arXiv:2501.00555*, 2024d.

Hongyi Wang, Kartik Sreenivasan, Shashank Rajput, Harit Vishwakarma, Saurabh Agarwal, Jy-yong Sohn, Kangwook Lee, and Dimitris Papailiopoulos. Attack of the tails: Yes, you really can backdoor federated learning. *Advances in neural information processing systems*, 33: 16070–16084, 2020a.

Ankit Pensia, Shashank Rajput, Alliot Nagle, Harit Vishwakarma, and Dimitris Papailiopoulos. Optimal lottery tickets via subset sum: Logarithmic over-parameterization is sufficient. *Advances in neural information processing systems*, 33:2599–2610, 2020.

Tzu-Heng Huang, Harit Vishwakarma, and Frederic Sala. Train'n trade: foundations of parameter markets. *Advances in Neural Information Processing Systems*, 36:28478–28490, 2023a.

Tzu-Heng Huang, Catherine Cao, Spencer Schoenberg, Harit Vishwakarma, Nicholas Roberts, and Frederic Sala. Scriptoriumws: A code generation assistant for weak supervision. In *ICLR Deep Learning for Code Workshop*, 2023b.

Changho Shin, Jitian Zhao, Sonia Cromp, Harit Vishwakarma, and Frederic Sala. OTTER: Effortless label distribution adaptation of zero-shot models. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024a.

Snorkel. Weak supervision at snorkel ai. https://snorkel.ai/weak-supervision/, 2021. Accessed: 2024-04-21.

SGT. Aws sagemaker ground truth. https://aws.amazon.com/sagemaker/data-labeling/, 2022. Accessed: 2022-11-18.

Ofer Dekel and Ohad Shamir. Vox populi: Collecting high-quality labels from a crowd. In *COLT*, 2009.

Vikas C. Raykar, Shipeng Yu, Linda H. Zhao, Gerardo Hermosillo Valadez, Charles Florin, Luca Bogoni, and Linda Moy. Learning from crowds. *Journal of Machine Learning Research*, 11(43): 1297–1322, 2010.

Ramya Korlakai Vinayak, Samet Oymak, and Babak Hassibi. Graph clustering with missing data: Convex algorithms and analysis. In *Advances in Neural Information Processing Systems*, 2014.

Yi Chen, Ramya Korlakai Vinayak, and Babak Hassibi. Crowdsourced clustering via active querying: Practical algorithm with theoretical guarantees. In *Proceedings of the AAAI Conference on Human Computation and Crowdsourcing*, volume 11, pages 27–37, 2023a.

Raymond Board and Leonard Pitt. Semi-supervised learning. *Machine Learning*, 4:41–65, 1989.

Xiaojin Zhu. Semi-supervised learning literature survey. In *University of Wisconsin-Madison, Department of Computer Sciences*, 2005.

Massih-Reza Amini, Vasilii Feofanov, Loic Pauletto, Lies Hadjadj, Emilie Devijver, and Yury Maximov. Self-training: A survey, 2023.

David Cohn, Les Atlas, and Richard Ladner. Improving generalization with active learning. *Machine Learning*, 15(2):201–221, 1994.

Maria-Florina Balcan, Andrei Z. Broder, and Tong Zhang. Margin based active learning. In *COLT*, 2007.

Yinglun Zhu and Robert Nowak. Efficient active learning with abstention. *arXiv preprint arXiv:2204.00043*, 2022.

Rich Caruana. Multitask learning. *Machine Learning*, 28:41–75, 1997. URL https://api.semanticscholar.org/CorpusID:45998148.

Lisa Torrey and Jude Shavlik. Transfer learning. In *Handbook of research on machine learning applications and trends: algorithms, methods, and techniques*, pages 242–264. IGI global, 2010.

Fuzhen Zhuang, Zhiyuan Qi, Keyu Duan, Dongbo Xi, Yongchun Zhu, Hengshu Zhu, Hui Xiong, and Qing He. A comprehensive survey on transfer learning. *Proceedings of the IEEE*, 109(1): 43–76, 2020.

Christopher M Bishop. Latent variable models. In *Learning in graphical models*, pages 371–403. Springer, 1998.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

Daniel Gildea and Daniel Jurafsky. Automatic labeling of semantic roles. *Computational Linguistics*, 28(3):245–288, 09 2002. ISSN 0891-2017.

Rahul S. Desikan, Florent Ségonne, Bruce Fischl, Brian T. Quinn, Bradford C. Dickerson, Deborah Blacker, Randy L. Buckner, Anders M. Dale, R. Paul Maguire, Bradley T. Hyman, Marilyn S. Albert, and Ronald J. Killiany. An automated labeling system for subdividing the human cerebral cortex on mri scans into gyral based regions of interest. *NeuroImage*, 31(3):968–980, 2006. ISSN 1053-8119.

Rishabh Mehrotra, Scott Sanner, Wray Buntine, and Lexing Xie. Improving lda topic models for microblogs via tweet pooling and automatic labeling. In *Proceedings of the 36th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '13, page 889–892. Association for Computing Machinery, 2013.

Chen Shi, Qi Chen, Lei Sha, Sujian Li, Xu Sun, Houfeng Wang, and Lintao Zhang. Auto-dialabel: Labeling dialogue data with unsupervised learning. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 684–689, Brussels, Belgium, October-November 2018. Association for Computational Linguistics.

Shikun Zhang, Omid Jafari, and Parth Nagarkar. A survey on machine learning techniques for auto labeling of video, audio, and text data, 2021a. URL https://arxiv.org/abs/2109.03784.

Yuki Markus Asano, Christian Rupprecht, and Andrea Vedaldi. Self-labelling via simultaneous clustering and representation learning, 2020.

Michael Desmond, Evelyn Duesterwald, Kristina Brimijoin, Michelle Brachman, and Qian Pan. Semi-automated data labeling. In *Proceedings of the NeurIPS 2020 Competition and Demonstration Track*, volume 133 of *Proceedings of Machine Learning Research*, pages 156–169. PMLR, 06–12 Dec 2021.

Hang Qiu, Krishna Chintalapudi, and Ramesh Govindan. MCAL: Minimum cost human-machine active labeling. In *The Eleventh International Conference on Learning Representations*, 2023.

A. J. Ratner, Christopher M. De Sa, Sen Wu, Daniel Selsam, and C. Ré. Data programming: Creating large training sets, quickly. In *Proceedings of the 29th Conference on Neural Information Processing Systems (NIPS 2016)*, Barcelona, Spain, 2016.

P. Varma, F. Sala, A. He, A. J. Ratner, and C. Ré. Learning dependency structures for weak supervision models. In *Proceedings of the 36th International Conference on Machine Learning (ICML 2019)*, 2019.

Daniel Y. Fu, Mayee F. Chen, Frederic Sala, Sarah M. Hooper, Kayvon Fatahalian, and Christopher Ré. Fast and three-rious: Speeding up weak supervision with triplet methods. In *Proceedings of the 37th International Conference on Machine Learning (ICML 2020)*, 2020.

Dong-Hyun Lee et al. Pseudo-label: The simple and efficient semi-supervised learning method for deep neural networks. 2013.

Ran El-Yaniv and Yair Wiener. On the foundations of noise-free selective classification. *JMLR*, 11:1605–1641, aug 2010. ISSN 1532-4435.

Martin J Wainwright and Michael I Jordan. Graphical models, exponential families, and variational inference. *Foundations and Trends® in Machine Learning*, 1(1-2):1–305, 2008.

Christopher M Bishop. Novelty detection and neural network validation. *IEE Proceedings-Vision, Image and Signal processing*, 141(4):217–222, 1994.

Kamal Nigam, Andrew Kachites McCallum, Sebastian Thrun, and Tom Mitchell. Text classification from labeled and unlabeled documents using em. *Machine learning*, 39:103–134, 2000.

Xiaojin Zhu, Zoubin Ghahramani, and John Lafferty. Semi-supervised learning using gaussian fields and harmonic functions. In *ICML*, 2003.

Zixing Song, Xiangli Yang, Zenglin Xu, and Irwin King. Graph-based semi-supervised learning: A comprehensive review. *arXiv*, 2021. arXiv:2102.13303.

Thorsten Joachims. Transductive inference for text classification using support vector machines. In Ivan Bratko and Saso Dzeroski, editors, *Proceedings of ICML-99, 16th International Conference on Machine Learning*, pages 200–209, 1999.

Kihyuk Sohn, David Berthelot, Nicholas Carlini, Zizhao Zhang, Han Zhang, Colin A Raffel, Ekin Dogus Cubuk, Alexey Kurakin, and Chun-Liang Li. Fixmatch: Simplifying semi-supervised learning with consistency and confidence. *Advances in neural information processing systems*, 33:596–608, 2020.

Yair Wiener and Ran El-Yaniv. Agnostic selective classification. *Advances in neural information processing systems*, 24, 2011.

Ran El-Yaniv and Yair Wiener. Active learning via perfect selective classification. *Journal of Machine Learning Research*, 13(2), 2012.

Yair Wiener and Ran El-Yaniv. Agnostic pointwise-competitive selective classification. *Journal of Artificial Intelligence Research*, 52:171–201, 2015.

Roei Gelbhart and Ran El-Yaniv. The relationship between agnostic selective classification, active learning and the disagreement coefficient. *The Journal of Machine Learning Research*, 20(1): 1136–1173, 2019.

Yonatan Geifman and Ran El-Yaniv. Selective classification for deep neural networks. In *Advances in Neural Information Processing Systems*, volume 30, 2017.

Olivier Gascuel and Gilles Caraux. Distribution-free performance bounds with the resubstitution error estimate. *Pattern Recognition Letters*, 13(11):757–764, 1992. ISSN 0167-8655.

Burr Settles. Active learning literature survey. 2009.

Steve Hanneke. Theory of disagreement-based active learning. *Found. Trends Mach. Learn.*, 7 (2–3):131–309, jun 2014. ISSN 1935-8237.

Pengzhen Ren, Yun Xiao, Xiaojun Chang, Po-Yao Huang, Zhihui Li, Brij B. Gupta, Xiaojiang Chen, and Xin Wang. A survey of deep active learning, 2020.

Matti Kääriäinen. Active learning in the non-realizable case. In *International Conference on Algorithmic Learning Theory*, pages 63–77. Springer, 2006.

Animashree Anandkumar, Rong Ge, Daniel Hsu, Sham M Kakade, and Matus Telgarsky. Tensor decompositions for learning latent variable models. *Journal of Machine Learning Research*, 15: 2773–2832, 2014a.

Alexander Philip Dawid and Allan M Skene. Maximum likelihood estimation of observer error-rates using the em algorithm. *Applied statistics*, pages 20–28, 1979.

Anh Nguyen, Jason Yosinski, and Jeff Clune. Deep neural networks are easily fooled: High confidence predictions for unrecognizable images. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 427–436, 2015a.

Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q Weinberger. On calibration of modern neural networks. In *International conference on machine learning*, pages 1321–1330. PMLR, 2017.

Ananya Kumar, Percy S Liang, and Tengyu Ma. Verified uncertainty calibration. *Advances in Neural Information Processing Systems*, 32, 2019.

Jooyoung Moon, Jihyo Kim, Younghak Shin, and Sangheum Hwang. Confidence-aware learning for deep neural networks. In *Proceedings of the 37th International Conference on Machine Learning*, volume 119, pages 7034–7044, 2020.

Chirag Gupta and Aaditya Ramdas. Top-label calibration and multiclass-to-binary reductions. In *International Conference on Learning Representations*, 2022.

Fei Zhu, Zhen Cheng, Xu-Yao Zhang, and Cheng-Lin Liu. Rethinking confidence calibration for failure prediction. In *European Conference on Computer Vision*, pages 518–536. Springer, 2022.

Like Hui, Mikhail Belkin, and Stephen Wright. Cut your losses with squentropy. In *Proceedings of the 40th International Conference on Machine Learning*, pages 14114–14131, 2023.

Bowen Zhang, Yidong Wang, Wenxin Hou, Hao Wu, Jindong Wang, Manabu Okumura, and Takahiro Shinozaki. Flexmatch: Boosting semi-supervised learning with curriculum pseudo labeling. *Advances in Neural Information Processing Systems*, 34:18408–18419, 2021b.

Charlotte Loh, Rumen Dangovski, Shivchander Sudalairaj, Seungwook Han, Ligong Han, Leonid Karlinsky, Marin Soljacic, and Akash Srivastava. On the importance of calibration in semi-supervised learning, 2022.

Shambhavi Mishra, Balamurali Murugesan, Ismail Ben Ayed, Marco Pedersoli, and Jose Dolz. Do not trust what you trust: Miscalibration in semi-supervised learning, 2024.

Changho Shin, Winfred Li, Harit Vishwakarma, Nicholas Carl Roberts, and Frederic Sala. Universalizing weak supervision. In *International Conference on Learning Representations*, 2022b.

Lev Goldfarb. A new approach to pattern recognition. *Progress in Pattern Recognition 2*, 234: 241–402, 1985.

Animashree Anandkumar, Rong Ge, and Majid Janzamin. Sample complexity analysis for learning overcomplete latent variable models through tensor methods. *arXiv preprint arXiv:1408.0553*, 2014b.

Carlo Ciliberto, Lorenzo Rosasco, and Alessandro Rudi. A consistent regularization approach for structured prediction. In *Advances in Neural Information Processing Systems 30 (NIPS 2016)*, volume 30, 2016.

Alexander Ratner, Stephen H. Bach, Henry Ehrenberg, Jason Fries, Sen Wu, and Christopher Ré. Snorkel: Rapid training data creation with weak supervision. In *Proceedings of the 44th International Conference on Very Large Data Bases (VLDB)*, Rio de Janeiro, Brazil, 2018.

Stephen H. Bach, Bryan He, Alexander J. Ratner, and Christopher Ré. Learning the structure of generative models without labeled data. In *Proceedings of the 34th International Conference on Machine Learning (ICML 2017)*, Sydney, Australia, 2017.

Frederic Sala, Paroma Varma, Jason Fries, Daniel Y. Fu, Shiori Sagawa, Saelig Khattar, Ashwini Ramamoorthy, Ke Xiao, Kayvon Fatahalian, James Priest, and Christopher Ré. Multi-resolution weak supervision for sequential data. In *Advances in Neural Information Processing Systems 32*, pages 192–203, 2019.

Sebastian Farquhar, Yarin Gal, and Tom Rainforth. On statistical bias in active learning: How and when to fix it. In *International Conference on Learning Representations*, 2020.

Kaidi Cao, Maria Brbic, and Jure Leskovec. Open-world semi-supervised learning. *arXiv preprint arXiv:2102.03526*, 2021.

Jingkang Yang, Pengyun Wang, Dejian Zou, Zitang Zhou, Kunyuan Ding, Wenxuan Peng, Haoqi Wang, Guangyao Chen, Bo Li, Yiyou Sun, Xuefeng Du, Kaiyang Zhou, Wayne Zhang, Dan Hendrycks, Yixuan Li, and Ziwei Liu. Openood: Benchmarking generalized out-of-distribution detection, 2022a.

Xinru Wang, Hannah Kim, Sajjadur Rahman, Kushan Mitra, and Zhengjie Miao. Human-llm collaborative annotation through effective verification of llm labels. In *Proceedings of the 2024 CHI Conference on Human Factors in Computing Systems*, CHI '24. Association for Computing Machinery, 2024.

Haodi Zhang, Junyu Yang, Jinyin Nie, Peirou Liang, Kaishun Wu, Defu Lian, Rui Mao, and Yuanfeng Song. Efficient data labeling by hierarchical crowdsourcing with large language models. In *Proceedings of the 31st International Conference on Computational Linguistics*, pages 11290–11303. Association for Computational Linguistics, 2025a.

Ziwei Ji, Nayeon Lee, Rita Frieske, Tiezheng Yu, Dan Su, Yan Xu, Etsuko Ishii, Ye Jin Bang, Andrea Madotto, and Pascale Fung. Survey of hallucination in natural language generation. *ACM computing surveys*, 55(12):1–38, 2023.

Adam Tauman Kalai and Santosh S Vempala. Calibrated language models must hallucinate. In *Proceedings of the 56th Annual ACM Symposium on Theory of Computing*, pages 160–171, 2024.

Charaka Vinayak Kumar, Ashok Urlana, Gopichand Kanumolu, Bala Mallikarjunarao Garlapati, and Pruthwik Mishra. No llm is free from bias: A comprehensive study of bias evaluation in large language models, 2025. URL https://arxiv.org/abs/2503.11985.

Tomas Horych, Christoph Mandl, Terry Ruas, Andre Greiner-Petter, Bela Gipp, Akiko Aizawa, and Timo Spinde. The promises and pitfalls of llm annotations in dataset labeling: a case study on media bias detection, 2025. URL https://arxiv.org/abs/2411.11081.

Joseph Redmon and Ali Farhadi. YOLO9000: better, faster, stronger. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6517–6525, 2017.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Proceedings of the 30th Conference on Neural Information Processing Systems (NIPS 2017)*, 2017.

Yufan Zeng and Jiashan Tang. Rlc-gnn: An improved deep architecture for spatial-based graph neural network with application to fraud detection. *Applied Sciences*, 11, 06 2021.

Ramya Korlakai Vinayak and Babak Hassibi. Crowdsourced clustering: Querying edges vs triangles. In *Proceedings of the 30th International Conference on Neural Information Processing Systems*, NIPS'16, 2016.

S. Fralick. Learning to recognize patterns without a teacher. *IEEE Transactions on Information Theory*, 13(1):57–64, 1967.

Superb-AI. Superb ai automated data labeling service. https://www.superb-ai.com/product/automate, 2022. Accessed: 2022-11-18.

Samsung-SDS. Samsung sds auto-labeling service. https://www.samsungsds.com/en/insights/TechToolkit_2021_Auto_Labeling.html , 2022. Accessed: 2022-11-18.

Airbus. Airbus active labeling blog. https://acubed.airbus.com/blog/wayfinder/automatic-data-labeling-strategies-for-vision-based-machine-learning-and-ai/ , 2022. Accessed: 2022-11-18.

Rudy Venguswamy, Mike Levy, Anirudh Koul, Satyarth Praveen, Tarun Narayanan, Ajay Krishnan, Jenessa Peterson, Siddha Ganju, and Meher Kasam. Curator: A No-Code Self-Supervised Learning and Active Labeling Tool to Create Labeled Image Datasets from Petabyte-Scale Imagery. In *EGU General Assembly Conference Abstracts*, pages EGU21–6853, April 2021.

William Feller. Generalization of a probability limit theorem of cramér. In *Transactions of the American Mathematical Society*, pages 361–372,, 1943.

Jannik Kossen, Sebastian Farquhar, Yarin Gal, and Tom Rainforth. Active testing: Sample-efficient model evaluation. *International Conference on Machine Learning*, 2021.

Ya Le and Xuan Yang. Tiny imagenet dataset. 2015a. Accessed: May 16, 2023.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pages 8748–8763. PMLR, 2021.

Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. Learning word vectors for sentiment analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 142–150, Portland, Oregon, USA, June 2011. Association for Computational Linguistics.

Shitao Xiao, Zheng Liu, Peitian Zhang, and Niklas Muennighoff. C-pack: Packaged resources to advance general chinese embedding, 2023.

Niklas Muennighoff, Nouamane Tazi, Loïc Magne, and Nils Reimers. Mteb: Massive text embedding benchmark. *arXiv preprint arXiv:2210.07316*, 2022.

Hugging Face. Hugging face mteb leaderboard. https://huggingface.co/spaces/mteb/leaderboard, 2023. Accessed: 2023-08-07.

Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009a.

Ting-Fan Wu, Chih-Jen Lin, and Ruby Weng. Probability estimates for multi-class classification by pairwise coupling. *Advances in Neural Information Processing Systems*, 16, 2003.

John Platt. Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. *Advances in large margin classifiers*, 10(3):61–74, 1999.

Sanjoy Dasgupta. Two faces of active learning. *Theoretical Computer Science*, 412(19):1767–1781, 2011. ISSN 0304-3975. Algorithmic Learning Theory (ALT 2009).

Daniel Joseph Hsu. *Algorithms for active learning*. PhD thesis, UC San Diego, 2010.

Gui Citovsky, Giulia DeSalvo, Claudio Gentile, Lazaros Karydas, Anand Rajagopalan, Afshin Rostamizadeh, and Sanjiv Kumar. Batch active learning at scale. In *Advances in Neural Information Processing Systems*, volume 34, pages 11933–11944, 2021.

Simon Tong and Daphne Koller. Support vector machine active learning with applications to text classification. *Journal of machine learning research*, 2(Nov):45–66, 2001.

Stephen Mussmann and Percy Liang. On the relationship between data efficiency and error for uncertainty sampling. In *Proceedings of the 35th International Conference on Machine Learning, ICML*, 2018.

Steve Hanneke. A bound on the label complexity of agnostic active learning. ICML, 2007.

Maria-Florina Balcan and Phil Long. Active and passive learning of linear separators under log-concave distributions. In *Conference on Learning Theory*, pages 288–316. PMLR, 2013.

Alina Beygelzimer, Sanjoy Dasgupta, and John Langford. Importance weighted active learning. In *Proceedings of the 26th annual international conference on machine learning*, pages 49–56, 2009.

Kamalika Chaudhuri, Sham M. Kakade, Praneeth Netrapalli, and Sujay Sanghavi. Convergence rates of active learning for maximum likelihood estimation. In *Proceedings of the 28th International Conference on Neural Information Processing Systems*, 2015.

Sanjoy Dasgupta, Adam Tauman Kalai, and Claire Monteleoni. Analysis of perceptron-based active learning. In *International conference on computational learning theory*, pages 249–263. Springer, 2005.

Sanjoy Dasgupta. Coarse sample complexity bounds for active learning. In Y. Weiss, B. Schölkopf, and J. Platt, editors, *Advances in Neural Information Processing Systems*, volume 18. MIT Press, 2006.

Akshay Krishnamurthy, Alekh Agarwal, Tzu-Kuo Huang, Hal Daumé III, and John Langford. Active learning for cost-sensitive classification. In *International Conference on Machine Learning*, pages 1915–1924. PMLR, 2017.

Julian Katz-Samuels, Jifan Zhang, Lalit Jain, and Kevin Jamieson. Improved algorithms for agnostic pool-based active classification. In *International Conference on Machine Learning*, pages 5334–5344. PMLR, 2021.

C Chow. On optimum recognition error and reject tradeoff. *IEEE Transactions on information theory*, 16(1):41–46, 1970.

Shubhanshu Shekhar, Mohammad Ghavamzadeh, and Tara Javidi. Active learning for classification with abstention. *IEEE Journal on Selected Areas in Information Theory*, 2(2):705–719, 2021. doi: 10.1109/JSAIT.2021.3081433.

Kulin Shah and Naresh Manwani. Online active learning of reject option classifiers. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(04):5652–5659, 2020.

Nikita Puchkin and Nikita Zhivotovskiy. Exponential savings in agnostic active learning through abstention. Proceedings of Machine Learning Research, pages 3806–3832. PMLR, 2021.

Corinna Cortes, Giulia DeSalvo, and Mehryar Mohri. Learning with rejection. In *International Conference on Algorithmic Learning Theory*, pages 67–82. Springer, 2016.

Matthias Hein, Maksym Andriushchenko, and Julian Bitterwolf. Why relu networks yield high-confidence predictions far away from the training data and how to mitigate the problem. 2018.

Jakob Gawlikowski, Cedrique Rovile Njieutcheu Tassi, Mohsin Ali, Jongseok Lee, Matthias Humt, Jianxiang Feng, Anna Kruspe, Rudolph Triebel, Peter Jung, Ribana Roscher, et al. A survey of uncertainty in deep neural networks. *arXiv preprint arXiv:2107.03342*, 2021a.

Matthias Minderer, Josip Djolonga, Rob Romijnders, Frances Hubis, Xiaohua Zhai, Neil Houlsby, Dustin Tran, and Mario Lucic. Revisiting the calibration of modern neural networks. In *Advances in Neural Information Processing Systems*, volume 34, pages 15682–15694, 2021.

Deng-Bao Wang, Lei Feng, and Min-Ling Zhang. Rethinking calibration of deep neural networks: Do not be afraid of overconfidence. In *Advances in Neural Information Processing Systems*, volume 34, pages 11809–11820, 2021a.

Ranganath Krishnan and Omesh Tickoo. Improving model calibration with accuracy versus uncertainty optimization. *Advances in Neural Information Processing Systems*, 33:18237–18248, 2020.

Chunwei Ma, Ziyun Huang, Jiayi Xian, Mingchen Gao, and Jinhui Xu. Improving uncertainty calibration of deep neural networks via truth discovery and geometric optimization. In *Proceedings of the Thirty-Seventh Conference on Uncertainty in Artificial Intelligence*, volume 161 of *Proceedings of Machine Learning Research*, pages 75–85. PMLR, 27–30 Jul 2021.

Nabeel Seedat and Christopher Kanan. Towards calibrated and scalable uncertainty representations for neural networks. *arXiv preprint arXiv:1911.00104*, 2019.

Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. In *International Conference on Learning Representations*, 2014.

Dan Hendrycks and Kevin Gimpel. A baseline for detecting misclassified and out-of-distribution examples in neural networks. In *International Conference on Learning Representations*, 2017a.

Yu Bai, Song Mei, Huan Wang, and Caiming Xiong. Don't just blame over-parametrization for over-confidence: Theoretical analysis of calibration in binary classification. In Marina Meila and Tong Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 566–576. PMLR, 18–24 Jul 2021.

Charles Corbière, Nicolas THOME, Avner Bar-Hen, Matthieu Cord, and Patrick Pérez. Addressing failure prediction by learning model confidence. In *Advances in Neural Information Processing Systems 32*, pages 2902–2913, 2019.

Shadab Hussain. Cifar 10- cnn using pytorch, Jul 2021. URL https://www.kaggle.com/code/shadabhussain/cifar-10-cnn-using-pytorch.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Shun-ichi Amari. Backpropagation and stochastic gradient descent method. *Neurocomputing*, 5 (4-5):185–196, 1993.

Léon Bottou. *Stochastic Gradient Descent Tricks*, pages 421–436. Springer Berlin Heidelberg, 2012. ISBN 978-3-642-35289-8.

Pierre Foret, Ariel Kleiner, Hossein Mobahi, and Behnam Neyshabur. Sharpness-aware minimization for efficiently improving generalization. In *International Conference on Learning Representations*, 2021.

Bianca Zadrozny and Charles Elkan. Transforming classifier scores into accurate multiclass probability estimates. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 694–699, 2002.

Meelis Kull, Miquel Perello Nieto, Markus Kängsepp, Telmo Silva Filho, Hao Song, and Peter Flach. Beyond temperature scaling: Obtaining well-calibrated multi-class probabilities with dirichlet calibration. In *Advances in Neural Information Processing Systems*, volume 32, 2019.

Tom Joy, Francesco Pinto, Ser-Nam Lim, Philip HS Torr, and Puneet K Dokania. Sample-dependent adaptive temperature scaling for improved calibration. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pages 14919–14926, 2023.

Yann LeCun. The mnist database of handwritten digits. *http://yann. lecun. com/exdb/mnist/*, 1998.

Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

Ya Le and Xuan Yang. Tiny imagenet visual recognition challenge. *CS 231N*, 7(7):3, 2015b.

Tom Mitchell. Twenty Newsgroups. UCI Machine Learning Repository, 1999.

David Lowell, Zachary C. Lipton, and Byron C. Wallace. Practical obstacles to deploying active learning. In Kentaro Inui, Jing Jiang, Vincent Ng, and Xiaojun Wan, editors, *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 21–30, Hong Kong, China, November 2019. Association for Computational Linguistics. doi: 10.18653/v1/D19-1003.

Alexander Sorokin and David Forsyth. Utility data annotation with amazon mechanical turk. In *2008 IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*, pages 1–8, 2008. doi: 10.1109/CVPRW.2008.4562953.

Ryan G. Gomes, Peter Welinder, Andreas Krause, and Pietro Perona. Crowdclustering. In *Advances in Neural Information Processing Systems 24*, pages 558–566. 2011.

David R Karger, Sewoong Oh, and Devavrat Shah. Budget-optimal crowdsourcing using low-rank matrix approximations. In *2011 49th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, pages 284–291. IEEE, 2011.

Arya Mazumdar and Barna Saha. Clustering with noisy queries. In *Advances in Neural Information Processing Systems*, volume 30, 2017.

Ramya Korlakai Vinayak, Tijana Zrnic, and Babak Hassibi. Tensor-based crowdsourced clustering via triangle queries. In *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 2322–2326. IEEE, 2017.

Changho Shin, Sonia Cromp, Dyah Adila, and Frederic Sala. Mitigating source bias for fairer weak supervision. In *Advances in Neural Information Processing Systems*, volume 36, pages 33564–33605, 2023.

Yue Yu, Simiao Zuo, Haoming Jiang, Wendi Ren, Tuo Zhao, and Chao Zhang. Fine-tuning pre-trained language model with weak supervision: A contrastive-regularized self-training approach. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1063–1077, 2021.

Dawei Zhu, Xiaoyu Shen, Marius Mosbach, Andreas Stephan, and Dietrich Klakow. Weaker than you think: A critical look at weakly supervised learning. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 14229–14253, 2023.

Jakob Gawlikowski, Cedrique Rovile Njieutcheu Tassi, Mohsin Ali, Jongseok Lee, Matthias Humt, Jianxiang Feng, Anna Kruspe, Rudolph Triebel, Peter Jung, Ribana Roscher, et al. A survey of uncertainty in deep neural networks. *arXiv preprint arXiv:2107.03342*, 2021b.

Anders Krogh and John A. Hertz. A simple weight decay can improve generalization. In *Proceedings of the 4th International Conference on Neural Information Processing Systems*, NIPS'91, page 950–957, San Francisco, CA, USA, 1991. Morgan Kaufmann Publishers Inc. ISBN 1558602224.

Bianca Zadrozny and Charles Elkan. Learning and making decisions when costs and probabilities are both unknown. In *Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 204–213, 2001.

Alexandru Niculescu-Mizil and Rich Caruana. Predicting good probabilities with supervised learning. In *Proceedings of the 22nd International Conference on Machine Learning*, ICML '05, page 625–632, 2005. ISBN 1595931805.

Aviral Kumar, Sunita Sarawagi, and Ujjwal Jain. Trainable calibration measures for neural networks from kernel mean embeddings. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 2805–2814. PMLR, 10–15 Jul 2018.

Rafael Müller, Simon Kornblith, and Geoffrey E Hinton. When does label smoothing help? *Advances in neural information processing systems*, 32, 2019.

Jishnu Mukhoti, Viveka Kulharia, Amartya Sanyal, Stuart Golodetz, Philip Torr, and Puneet Dokania. Calibrating deep neural networks using focal loss. *Advances in Neural Information Processing Systems*, 33:15288–15299, 2020.

Chirag Gupta and Aaditya Ramdas. Distribution-free calibration guarantees for histogram binning without sample splitting. In *International Conference on Machine Learning*, pages 3942–3952. PMLR, 2021.

Olivier Chapelle, Bernhard Schölkopf, and Alexander Zien, editors. *Semi-Supervised Learning*. The MIT Press, 2006. ISBN 9780262033589.

Jesper E. van Engelen and Holger H. Hoos. A survey on semi-supervised learning. *Machine Learning*, 109:373 – 440, 2019.

Geoffrey J McLachlan. Iterative reclassification procedure for constructing an asymptotically optimal rule of allocation in discriminant analysis. *Journal of the American Statistical Association*, 70(350):365–369, 1975.

Chuck Rosenberg, Martial Hebert, and Henry Schneiderman. Semi-supervised self-training of object detection models. In *Seventh IEEE Workshops on Applications of Computer Vision (WACV/MOTION'05) - Volume 1*, volume 1, pages 29–36, 2005. doi: 10.1109/ACVMOT.2005.107.

Mamshad Nayeem Rizve, Kevin Duarte, Yogesh S Rawat, and Mubarak Shah. In defense of pseudo-labeling: An uncertainty-aware pseudo-label selection framework for semi-supervised learning. In *International Conference on Learning Representations*, 2021.

Samuli Laine and Timo Aila. Temporal ensembling for semi-supervised learning. *Fifth International Conference on Learning Representations*, 2017.

Philip Bachman, Ouais Alsharif, and Doina Precup. Learning with pseudo-ensembles. In *Advances in Neural Information Processing Systems*, volume 27, 2014.

Mehdi Sajjadi, Mehran Javanmardi, and Tolga Tasdizen. Regularization with stochastic transformations and perturbations for deep semi-supervised learning. In *Proceedings of the 30th International Conference on Neural Information Processing Systems*, page 1171–1179, 2016.

Yue Fan, Anna Kukleva, and Bernt Schiele. Revisiting consistency regularization for semi-supervised learning, 2021.

Jan Kukačka, Vladimir Golkov, and Daniel Cremers. Regularization for deep learning: A taxonomy, 2017.

Yidong Wang, Hao Chen, Qiang Heng, Wenxin Hou, Yue Fan, Zhen Wu, Jindong Wang, Marios Savvides, Takahiro Shinozaki, Bhiksha Raj, Bernt Schiele, and Xing Xie. Freematch: Self-adaptive thresholding for semi-supervised learning. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=PDrUPTXJI_A.

Yi Xu, Lei Shang, Jinxing Ye, Qi Qian, Yu-Feng Li, Baigui Sun, Hao Li, and Rong Jin. Dash: Semi-supervised learning with dynamic thresholding. In *International Conference on Machine Learning*, pages 11525–11536. PMLR, 2021.

Hao Chen, Ran Tao, Yue Fan, Yidong Wang, Jindong Wang, Bernt Schiele, Xing Xie, Bhiksha Raj, and Marios Savvides. Softmatch: Addressing the quantity-quality tradeoff in semi-supervised

learning. In *The Eleventh International Conference on Learning Representations*, 2023b. URL https://openreview.net/forum?id=ymt1zQXBDiF.

Muyang Li, Runze Wu, Haoyu Liu, Jun Yu, Xun Yang, Bo Han, and Tongliang Liu. Instant: Semi-supervised learning with instance-dependent thresholds. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023.

Andrija Djurisic, Nebojsa Bozanic, Arjun Ashok, and Rosanne Liu. Extremely simple activation shaping for out-of-distribution detection. In *The Eleventh International Conference on Learning Representations*, 2023.

Charlotte Loh, Rumen Dangovski, Shivchander Sudalairaj, Seungwook Han, Ligong Han, Leonid Karlinsky, Marin Soljacic, and Akash Srivastava. Mitigating confirmation bias in semi-supervised learning via efficient bayesian model averaging. *Transactions on Machine Learning Research*, 2023.

Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009b.

Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Baolin Wu, Andrew Y Ng, et al. Reading digits in natural images with unsupervised feature learning. In *NIPS workshop on deep learning and unsupervised feature learning*, volume 2011, page 7. Granada, Spain, 2011.

Yidong Wang, Hao Chen, Yue Fan, Wang Sun, Ran Tao, Wenxin Hou, Renjie Wang, Linyi Yang, Zhi Zhou, Lan-Zhe Guo, Heli Qi, Zhen Wu, Yu-Feng Li, Satoshi Nakamura, Wei Ye, Marios Savvides, Bhiksha Raj, Takahiro Shinozaki, Bernt Schiele, Jindong Wang, Xing Xie, and Yue Zhang. Usb: A unified semi-supervised learning benchmark for classification. In *Thirty-sixth Conference on Neural Information Processing Systems, Datasets and Benchmarks Track*, 2022a.

Aarti Singh, Robert Nowak, and Jerry Zhu. Unlabeled data: Now it helps, now it doesn't. In *Advances in Neural Information Processing Systems*, volume 21. Curran Associates, Inc., 2008.

Avital Oliver, Augustus Odena, Colin A Raffel, Ekin Dogus Cubuk, and Ian Goodfellow. Realistic evaluation of deep semi-supervised learning algorithms. In *Advances in Neural Information Processing Systems*, volume 31, 2018.

Henry Scudder. Probability of error of some adaptive pattern-recognition machines. *IEEE Transactions on Information Theory*, 11(3):363–371, 1965.

Avrim Blum and Tom Mitchell. Combining labeled and unlabeled data with co-training. In *Proc. of the eleventh annual conference on Computational learning theory*, pages 92–100. ACM, 1998.

Samet Oymak and Talha Cihad Gulcu. Statistical and algorithmic insights for semi-supervised learning with self-training. *arXiv preprint arXiv:2006.11006*, 2020.

Ryan Prescott Adams and Zoubin Ghahramani. Archipelago: nonparametric bayesian semi-supervised learning. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pages 1–8, 2009.

Durk P Kingma, Shakir Mohamed, Danilo Jimenez Rezende, and Max Welling. Semi-supervised learning with deep generative models. In *Advances in Neural Information Processing Systems*, volume 27, 2014.

Avrim Blum and Shuchi Chawla. Learning from labeled and unlabeled data using graph mincuts. 2001.

Partha Niyogi. Manifold regularization and semi-supervised learning: Some theoretical analyses. *Journal of Machine Learning Research*, 14(5), 2013.

Amarnag Subramanya and Partha Pratim Talukdar. *Graph-based semi-supervised learning*. Springer Nature, 2022.

Vladimir Naumovich Vapnik, Vlamimir Vapnik, et al. Statistical learning theory. 1998.

Giannis Karamanolakis, Subhabrata Mukherjee, Guoqing Zheng, and Ahmed Hassan Awadallah. Self-training with weak supervision. *arXiv preprint arXiv:2104.05514*, 2021.

Jacob Kahn, Ann Lee, and Awni Hannun. Self-training for end-to-end speech recognition. In *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 7084–7088. IEEE, 2020.

Yasser El-Manzalawy, Elyse E Munoz, Scott E Lindner, and Vasant Honavar. Plasmosep: Predicting surface-exposed proteins on the malaria parasite using semisupervised self-training and expert-annotated data. *Proteomics*, 16(23):2967–2976, 2016.

Eric Arazo, Diego Ortego, Paul Albert, Noel E O'Connor, and Kevin McGuinness. Pseudo-labeling and confirmation bias in deep semi-supervised learning. In *2020 International joint conference on neural networks (IJCNN)*, pages 1–8. IEEE, 2020.

Qizhe Xie, Zihang Dai, Eduard Hovy, Thang Luong, and Quoc Le. Unsupervised data augmentation for consistency training. In *Advances in Neural Information Processing Systems*, volume 33, pages 6256–6268, 2020.

Dan Hendrycks and Kevin Gimpel. A baseline for detecting misclassified and out-of-distribution examples in neural networks. In *International Conference on Learning Representations*, 2017b.

Jared A. Dunnmon, Alexander J. Ratner, Khaled Saab, Nishith Khandwala, Matthew Markert, Hersh Sagreiya, Roger Goldman, Christopher Lee-Messer, Matthew P. Lungren, Daniel L. Rubin, and Christopher Ré. Cross-modal data programming enables rapid medical machine learning. *Patterns*, 1(2), 2020.

Christopher Ré, Feng Niu, Pallavi Gudipati, and Charles Srisuwananukorn. Overton: A data system for monitoring and improving machine-learned products. In *Proceedings of the 10th Annual Conference on Innovative Data Systems Research*, 2020.

Esteban Safranchik, Shiying Luo, and Stephen Bach. Weakly supervised sequence tagging from noisy rules. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, pages 5570–5578, Apr. 2020.

Gükhan H. Bakir, Thomas Hofmann, Bernhard Schölkopf, Alexander J. Smola, Ben Taskar, and S. V. N. Vishwanathan. *Predicting Structured Data (Neural Information Processing)*. The MIT Press, 2007.

Volodymyr Kuleshov and Percy S Liang. Calibrated structured prediction. In *Advances in Neural Information Processing Systems 28 (NIPS 2015)*, 2015.

Anna Korba and Florence d'Alché-Buc Alexandre Garcia. A structured prediction approach for label ranking. In *Advances in Neural Information Processing Systems 32 (NeurIPS 2018)*, volume 32, 2018.

Alexander Petersen and Hans-Georg Müller. Fréchet regression for random objects with euclidean predictors. *Annals of Statistics*, 47(2):691–719, 2019.

Alessandro Rudi, Carlo Ciliberto, GianMaria Marconi, and Lorenzo Rosasco. Manifold structured prediction. In *Advances in Neural Information Processing Systems 32 (NeurIPS 2018)*, volume 32, 2018.

Colin Graber and Alexander Schwing. Graph structured prediction energy networks. In *Advances in Neural Information Processing Systems 33 (NeurIPS 2019)*, volume 33, 2019.

A. J. Ratner, B. Hancock, J. Dunnmon, F. Sala, S. Pandey, and C. Ré. Training complex models with multi-task weak supervision. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Honolulu, Hawaii, 2019.

Nagarajan Natarajan, Inderjit S. Dhillon, Pradeep Ravikumar, and Ambuj Tewari. Learning with noisy labels. In *Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 1*, NIPS'13, page 1196–1204, 2013a.

J.B. Kruskal and M. Wish. *Multidimensional Scaling*. Sage Publications, 1978.

Elżbieta Pękalska, Pavel Paclik, and Robert P.W. Duin. A generalized kernel approach to dissimilarity-based classification. *Journal of Machine Learning Research*, 2:175–211, 2001.

Julian Laub, Volker Roth, Joachim M Buhmann, and Klaus-Robert Müller. On the information and representation of non-euclidean pairwise data. *Pattern Recognition*, 39(10):1815–1826, 2006.

Elżbieta Pękalska, Artsiom Harol, Robert P. W. Duin, Barbara Spillmann, and Horst Bunke. Non-euclidean or non-metric measures can be informative. In Dit-Yan Yeung, James T. Kwok, Ana Fred, Fabio Roli, and Dick de Ridder, editors, *Structural, Syntactic, and Statistical Pattern Recognition*, pages 871–880, 2006.

Brendan van Rooyen and Robert C. Williamson. A theory of learning with corrupted labels. *Journal of Machine Learning Research*, 18(228):1–50, 2018.

John M. Lee. *Introduction to Smooth Manifolds*. Springer, 2000.

Loring W. Tu. *An Introduction to Manifolds*. Springer, 2011.

Austin J. Stromme. *Wasserstein Barycenters: Statistics and Optimization*. MIT, 2020.

Helmer Aslaksen and Hsueh-Ling Huynh. Laws of trigonometry in symmetric spaces. *Geometry from the Pacific Rim*, 1991.

Dario Amodei, Chris Olah, Jacob Steinhardt, Paul F. Christiano, John Schulman, and Dan Mané. Concrete problems in AI safety. *CoRR*, abs/1606.06565, 2016.

Bernard Rosner. On the detection of many outliers. *Technometrics*, 17(2):221–227, 1975.

Douglas M Hawkins. Multivariate outlier detection. In *Identification of outliers*, pages 104–114. Springer, 1980.

G. Danuser and M. Stricker. Parametric model fitting: from inlier characterization to outlier detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(3):263–280, 1998.

Charu C. Aggarwal and Philip S. Yu. Outlier detection for high dimensional data. *SIGMOD Rec.*, 30(2):37–46, 2001. ISSN 0163-5808.

Varun Chandola, Arindam Banerjee, and Vipin Kumar. Anomaly detection: A survey. *ACM Comput. Surv.*, 41(3), July 2009a.

Varun Chandola, Arindam Banerjee, and Vipin Kumar. Anomaly detection: A survey. *ACM computing surveys (CSUR)*, 41(3):1–58, 2009b.

Anh Mai Nguyen, Jason Yosinski, and Jeff Clune. Deep neural networks are easily fooled: High confidence predictions for unrecognizable images. In *CVPR*, pages 427–436. IEEE Computer Society, 2015b.

Dan Hendrycks and Kevin Gimpel. A baseline for detecting misclassified and out-of-distribution examples in neural networks. *arXiv preprint arXiv:1610.02136*, 2016.

Mohammadreza Salehi, Hossein Mirzaei, Dan Hendrycks, Yixuan Li, Mohammad Hossein Rohban, and Mohammad Sabokrou. A unified survey on anomaly, novelty, open-set, and out-of-distribution detection: Solutions and future challenges. In *Transactions on Machine Learning Research*, 2022.

Jingkang Yang, Kaiyang Zhou, Yixuan Li, and Ziwei Liu. Generalized out-of-distribution detection: A survey, 2022b.

Shiyu Liang, Yixuan Li, and R. Srikant. Enhancing the reliability of out-of-distribution image detection in neural networks. In *International Conference on Learning Representations*, 2018.

Weitang Liu, Xiaoyun Wang, John Owens, and Yixuan Li. Energy-based out-of-distribution detection. *Advances in Neural Information Processing Systems*, 33:21464–21475, 2020.

Yiyou Sun, Chuan Guo, and Yixuan Li. React: Out-of-distribution detection with rectified activations. *Advances in Neural Information Processing Systems*, 34:144–157, 2021.

Sima Behpour, Thang Doan, Xin Li, Wenbin He, Liang Gou, and Liu Ren. Gradorth: A simple yet efficient out-of-distribution detection with orthogonal projection of gradients, 2023.

Yifei Ming, Yiyou Sun, Ousmane Dia, and Yixuan Li. Cider: Exploiting hyperspherical embeddings for out-of-distribution detection. *arXiv preprint arXiv:2203.04450*, 2022.

Dan Hendrycks, Mantas Mazeika, and Thomas Dietterich. Deep anomaly detection with outlier exposure. In *International Conference on Learning Representations*, 2019.

Petra Bevandić, Ivan Krešo, Marin Oršić, and Siniša Šegvić. Discriminative out-of-distribution detection for semantic segmentation, 2018.

Jingkang Yang, Haoqi Wang, Litong Feng, Xiaopeng Yan, Huabin Zheng, Wayne Zhang, and Ziwei Liu. Semantically coherent out-of-distribution detection. *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 8281–8289, 2021a.

Julian Katz-Samuels, Julia B. Nakhleh, Robert D. Nowak, and Yixuan Li. Training ood detectors in their natural habitats. In *International Conference on Machine Learning*, 2022a.

Rikard Laxhammar and Göran Falkman. Sequential conformal anomaly detection in trajectories based on hausdorff distance. In *14th International Conference on Information Fusion*, pages 1–8, 2011.

Rikard Laxhammar and Göran Falkman. Inductive conformal anomaly detection for sequential detection of anomalous sub-trajectories. *Annals of Mathematics and Artificial Intelligence*, 74: 67–94, 2015.

Ramneet Kaur, Susmit Jha, Anirban Roy, Sangdon Park, Edgar Dobriban, Oleg Sokolsky, and Insup Lee. idecode: In-distribution equivariance for conformal out-of-distribution detection. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pages 7104–7114, 2022.

Stephen Bates, Emmanuel Candès, Lihua Lei, Yaniv Romano, and Matteo Sesia. Testing for outliers with conformal p-values. *The Annals of Statistics*, 51(1):149 – 178, 2023.

Marvin Zhang, Sergey Levine, and Chelsea Finn. Memo: Test time robustness via adaptation and augmentation. *Advances in Neural Information Processing Systems*, 35:38629–38642, 2022.

Ke Fan, Yikai Wang, Qian Yu, Da Li, and Yanwei Fu. A simple test-time method for out-of-distribution detection. *arXiv preprint arXiv:2207.08210*, 2022.

Dequan Wang, Evan Shelhamer, Shaoteng Liu, Bruno Olshausen, and Trevor Darrell. Tent: Fully test-time adaptation by entropy minimization. *arXiv preprint arXiv:2006.10726*, 2020b.

D. A. Darling and Herbert Robbins. Confidence sequences for mean, variance, and median. *Proceedings of the National Academy of Sciences*, 58(1):66–68, 1967.

Steven R. Howard and Aaditya Ramdas. Sequential estimation of quantiles with applications to A/B testing and best-arm identification. *Bernoulli*, 28(3):1704 – 1728, 2022. doi: 10.3150/21-BEJ1388.

Ramesh Johari. Can i take a peek?: Continuous monitoring of online a/b tests. *Proceedings of the 24th International Conference on World Wide Web*, 2015.

Kevin Jamieson, Matthew Malloy, Robert Nowak, and Sébastien Bubeck. lil' ucb : An optimal exploration algorithm for multi-armed bandits. In *Proceedings of The 27th Conference on Learning Theory*, volume 35, pages 423–439. PMLR, 2014.

Balazs Szorenyi, Robert Busa-Fekete, Paul Weng, and Eyke Hüllermeier. Qualitative multi-armed bandits: A quantile-based approach. In *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pages 1660–1668, Lille, France, 07–09 Jul 2015. PMLR.

Akshay Balsubramani. Sharp finite-time iterated-logarithm martingale concentration, 2015.

Christopher Mohri, Daniel Andor, Eunsol Choi, Michael Collins, Anqi Mao, and Yutao Zhong. Learning to reject with a fixed predictor: Application to decontextualization. In *The Twelfth International Conference on Learning Representations*, 2024.

Mingda Qiao and Gregory Valiant. A theory of selective prediction, 2019. URL https://arxiv.org/abs/1902.04256.

Aditya Gangrade, Anil Kag, and Venkatesh Saligrama. Selective classification via one-sided prediction. In *Proceedings of The 24th International Conference on Artificial Intelligence and Statistics*. PMLR, 13–15 Apr 2021.

David Madras, Toni Pitassi, and Richard Zemel. Predict responsibly: improving fairness and accuracy by learning to defer. *Advances in neural information processing systems*, 31, 2018.

Dharmesh Tailor, Aditya Patra, Rajeev Verma, Putra Manggala, and Eric Nalisnick. Learning to defer to a population: A meta-learning approach. In *Proceedings of The 27th International Conference on Artificial Intelligence and Statistics*, volume 238 of *Proceedings of Machine Learning Research*, pages 3475–3483. PMLR, 02–04 May 2024a.

Rajeev Verma and Eric Nalisnick. Calibrated learning to defer with one-vs-all classifiers. In *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pages 22184–22202. PMLR, 17–23 Jul 2022.

David H Wolpert. Stacked generalization. *Neural networks*, 5(2):241–259, 1992.

Giulia DeSalvo, Mehryar Mohri, and Umar Syed. Learning with deep cascades. In *Proceedings of the Twenty-Sixth International Conference on Algorithmic Learning Theory (ALT 2015)*, 2015.

Wittawat Jitkrittum, Neha Gupta, Aditya K Menon, Harikrishna Narasimhan, Ankit Rawat, and Sanjiv Kumar. When does confidence-based cascade deferral suffice? In *Advances in Neural Information Processing Systems*, volume 36, pages 9891–9906, 2023.

Neha Gupta, Harikrishna Narasimhan, Wittawat Jitkrittum, Ankit Singh Rawat, Aditya Krishna Menon, and Sanjiv Kumar. Language model cascades: Token-level uncertainty and beyond, 2024a. URL https://arxiv.org/abs/2404.10136.

Divya Shanmugam, Davis Blalock, Guha Balakrishnan, and John Guttag. Better aggregation in test-time augmentation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 1214–1223, October 2021.

Yongqin Xian, Zeynep Akata, Gaurav Sharma, Quynh Nguyen, Matthias Hein, and Bernt Schiele. Latent embeddings for zero-shot classification. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 69–77, 2016.

Wei Wang, Vincent W Zheng, Han Yu, and Chunyan Miao. A survey of zero-shot learning: Settings, methods, and applications. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 10(2):1–37, 2019.

Changho Shin, Jitian Zhao, Sonia Cromp, Harit Vishwakarma, and Frederic Sala. Otter: Effortless label distribution adaptation of zero-shot models, 2024b. URL https://arxiv.org/abs/2404.08461.

Dyah Adila, Changho Shin, Linrong Cai, and Frederic Sala. Zero-shot robustification of zero-shot models, 2024. URL https://arxiv.org/abs/2309.04344.

Charlie Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. Scaling llm test-time compute optimally can be more effective than scaling model parameters. *arXiv preprint arXiv:2408.03314*, 2024.

Zhe Chen, Weiyun Wang, Yue Cao, Yangzhou Liu, Zhangwei Gao, Erfei Cui, Jinguo Zhu, Shenglong Ye, Hao Tian, Zhaoyang Liu, et al. Expanding performance boundaries of open-source multimodal models with model, data, and test-time scaling. *arXiv preprint arXiv:2412.05271*, 2024.

Qiyuan Zhang, Fuyuan Lyu, Zexu Sun, Lei Wang, Weixu Zhang, Zhihan Guo, Yufei Wang, Irwin King, Xue Liu, and Chen Ma. What, how, where, and how well? a survey on test-time scaling in large language models. *arXiv preprint arXiv:2503.24235*, 2025b.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837, 2022a.

Pengfei Liu, Weizhe Yuan, Jinlan Fu, Zhengbao Jiang, Hiroaki Hayashi, and Graham Neubig. Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing. *ACM computing surveys*, 55(9):1–35, 2023.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models. *arXiv preprint arXiv:2203.11171*, 2022b.

Vladimir Vovk, Alex Gammerman, and Glenn Shafer. *Algorithmic Learning in a Random World*. Springer-Verlag, Berlin, Heidelberg, 2005a. ISBN 0387001522.

Changle Qu, Sunhao Dai, Xiaochi Wei, Hengyi Cai, Shuaiqiang Wang, Dawei Yin, Jun Xu, and Ji-Rong Wen. Tool learning with large language models: A survey. *arXiv preprint arXiv:2405.17935*, 2024.

Bhawesh Kumar, Charlie Lu, Gauri Gupta, Anil Palepu, David Bellamy, Ramesh Raskar, and Andrew Beam. Conformal prediction with large language models for multi-choice question answering. *arXiv preprint arXiv:2305.18404*, 2023.

Jiayuan Su, Jing Luo, Hongwei Wang, and Lu Cheng. Api is enough: Conformal prediction for large language models without logit-access. *arXiv preprint arXiv:2403.01216*, 2024.

Jeffrey S. Rosenthal. Monty hall, monty fall, monty crawl. *Math Horizons*, 16(1), 2008.

Nicolas Schreuder, Victor-Emmanuel Brunel, and Arnak Dalalyan. A nonasymptotic law of iterated logarithm for general m-estimators. In *Proceedings of the Twenty Third International Conference on Artificial Intelligence and Statistics*, volume 108 of *Proceedings of Machine Learning Research*, pages 1331–1341. PMLR, 26–28 Aug 2020.

Ramesh Johari, Leo Pekelis, and David J. Walsh. Always valid inference: Bringing sequential analysis to a/b testing, 2019. URL https://arxiv.org/abs/1512.04922.

A. Dvoretzky, J. Kiefer, and J. Wolfowitz. Asymptotic Minimax Character of the Sample Distribution Function and of the Classical Multinomial Estimator. *The Annals of Mathematical Statistics*, 27(3):642 – 669, 1956.

Pascal Massart. The tight constant in the dvoretzky-kiefer-wolfowitz inequality. *The annals of Probability*, pages 1269–1283, 1990a.

Vladimir Vovk, Alexander Gammerman, and Glenn Shafer. *Algorithmic learning in a random world*, volume 29. Springer, 2005b.

Glenn Shafer and Vladimir Vovk. A tutorial on conformal prediction. *Journal of Machine Learning Research*, 9(12):371–421, 2008.

Anastasios N. Angelopoulos and Stephen Bates. A Gentle Introduction to Conformal Prediction and Distribution-Free Uncertainty Quantification. *arXiv preprint arXiv:2107.07511*, (arXiv:2107.07511), 2022.

Christopher Mohri and Tatsunori Hashimoto. Language models with conformal factuality guarantees. *arXiv preprint arXiv:2402.10978*, 2024.

John J. Cherian, Isaac Gibbs, and Emmanuel J. Candès. Large language model validity via enhanced conformal prediction methods. *arXiv preprint arXiv:2406.09714*, 2024.

Shiyu Liang, Yixuan Li, and Rayadurgam Srikant. Enhancing the reliability of out-of-distribution image detection in neural networks. *arXiv preprint arXiv:1706.02690*, 2017.

Kimin Lee, Kibok Lee, Honglak Lee, and Jinwoo Shin. A simple unified framework for detecting out-of-distribution samples and adversarial attacks. *Advances in neural information processing systems*, 31, 2018a.

Jingkang Yang, Kaiyang Zhou, Yixuan Li, and Ziwei Liu. Generalized out-of-distribution detection: A survey. *arXiv preprint arXiv:2110.11334*, 2021b.

Aleksandr Khinchine. Über einen satz der wahrscheinlichkeitsrechnung. *Fundamenta Mathematicae*, 6:9–20, 1924.

Wassily Hoeffding. Probability inequalities for sums of bounded random variables. *Journal of the American Statistical Association*, 58(301), 1963. ISSN 01621459.

Vikash Sehwag, Mung Chiang, and Prateek Mittal. Ssd: A unified framework for self-supervised outlier detection. *arXiv preprint arXiv:2103.12051*, 2021.

Haoqi Wang, Zhizhong Li, Litong Feng, and Wayne Zhang. Vim: Out-of-distribution with virtual-logit matching. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4921–4930, 2022c.

Yiyou Sun, Yifei Ming, Xiaojin Zhu, and Yixuan Li. Out-of-distribution detection with deep nearest neighbors. In *International Conference on Machine Learning*, pages 20827–20840. PMLR, 2022.

Abhijit Bendale and Terrance E. Boult. Towards open set deep networks. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1563–1572, 2015.

Haoran Wang, Weitang Liu, Alex E. Bocchieri, and Yixuan Li. Can multi-label classification networks know what they don't know? In *Neural Information Processing Systems*, 2021b.

Qitian Wu, Yiting Chen, Chenxiao Yang, and Junchi Yan. Energy-based out-of-distribution detection for graph neural networks. In *The Eleventh International Conference on Learning Representations*, 2023.

Rui Huang, Andrew Geng, and Yixuan Li. On the importance of gradients for detecting distributional shifts in the wild. *Advances in Neural Information Processing Systems*, 34: 677–689, 2021.

Jie Ren, Jiaming Luo, Yao Zhao, Kundan Krishna, Mohammad Saleh, Balaji Lakshminarayanan, and Peter J. Liu. Out-of-distribution detection and selective generation for conditional language models. In *The Eleventh International Conference on Learning Representations,ICLR 2023*, 2023a.

Jihoon Tack, Sangwoo Mo, Jongheon Jeong, and Jinwoo Shin. Csi: Novelty detection via contrastive learning on distributionally shifted instances. In *Advances in Neural Information Processing Systems*, volume 33, pages 11839–11852. Curran Associates, Inc., 2020.

Eric Nalisnick, Akihiro Matsukawa, Yee Whye Teh, Dilan Gorur, and Balaji Lakshminarayanan. Hybrid models with deep and invertible features. In *International Conference on Machine Learning*, pages 4723–4732. PMLR, 2019.

Jie Ren, Peter J. Liu, Emily Fertig, Jasper Snoek, Ryan Poplin, Mark Depristo, Joshua Dillon, and Balaji Lakshminarayanan. Likelihood ratios for out-of-distribution detection. In *Advances in Neural Information Processing Systems*, volume 32, 2019.

Polina Kirichenko, Pavel Izmailov, and Andrew G Wilson. Why normalizing flows fail to detect out-of-distribution data. *Advances in neural information processing systems*, 33:20578–20589, 2020.

Robin Schirrmeister, Yuxuan Zhou, Tonio Ball, and Dan Zhang. Understanding anomaly detection with deep invertible networks through hierarchies of distributions and features. *Advances in Neural Information Processing Systems*, 33:21038–21049, 2020.

Joan Serrà, David Álvarez, Vicenç Gómez, Olga Slizovskaia, José F. Núñez, and Jordi Luque. Input complexity and out-of-distribution detection with likelihood-based generative models. In *International Conference on Learning Representations*, 2020.

Jim Winkens, Rudy Bunel, Abhijit Guha Roy, Robert Stanforth, Vivek Natarajan, Joseph R. Ledsam, Patricia MacWilliams, Pushmeet Kohli, Alan Karthikesalingam, Simon A. A. Kohl, taylan. cemgil, S. M. Ali Eslami, and Olaf Ronneberger. Contrastive training for improved out-of-distribution detection. *ArXiv*, abs/2007.05566, 2020.

Zhisheng Xiao, Qing Yan, and Yali Amit. Likelihood regret: An out-of-distribution detection score for variational auto-encoder. In *Advances in Neural Information Processing Systems*, volume 33, pages 20685–20696. Curran Associates, Inc., 2020.

Yonatan Geifman and Ran El-Yaniv. SelectiveNet: A deep neural network with an integrated reject option. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97, pages 2151–2159, 2019.

Sina Mohseni, Mandar Pitale, Jbs Yadawa, and Zhangyang Wang. Self-supervised learning for generalizable out-of-distribution detection. In *AAAI Conference on Artificial Intelligence*, 2020.

Taewon Jeong and Heeyoung Kim. Ood-maml: Meta-learning for few-shot out-of-distribution detection and classification. In *Advances in Neural Information Processing Systems*, volume 33, pages 3907–3916, 2020.

Hongxin Wei, Renchunzi Xie, Hao Cheng, Lei Feng, Bo An, and Yixuan Li. Mitigating neural network overconfidence with logit normalization. In *Proceedings of the 39th International Conference on Machine Learning*, volume 162, pages 23631–23644, 2022b.

Kimin Lee, Honglak Lee, Kibok Lee, and Jinwoo Shin. Training confidence-calibrated classifiers for detecting out-of-distribution samples. 2018b.

Si Liu, Risheek Garrepalli, Thomas Dietterich, Alan Fern, and Dan Hendrycks. Open category detection with PAC guarantees. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80, pages 3169–3178. PMLR, 2018.

Nicolas Papernot and Patrick Mcdaniel. Deep k-nearest neighbors: Towards confident, interpretable and robust deep learning. *ArXiv*, abs/1803.04765, 2018.

Feiyang Cai and Xenofon Koutsoukos. Real-time out-of-distribution detection in learning-enabled cyber-physical systems. In *2020 ACM/IEEE 11th International Conference on Cyber-Physical Systems (ICCPS)*, pages 174–183, 2020.

Yusuke Iwasawa and Yutaka Matsuo. Test-time classifier adjustment module for model-agnostic domain generalization. *Advances in Neural Information Processing Systems*, 34:2427–2440, 2021.

Charu C. Aggarwal. *An Introduction to Outlier Analysis*, pages 1–34. Springer International Publishing, Cham, 2017. ISBN 978-3-319-47578-3.

Tze Leung Lai. On Confidence Sequences. *The Annals of Statistics*, 4(2):265 – 280, 1976.

Akash Maharaj, Ritwik Sinha, David Arbour, Ian Waudby-Smith, Simon Z. Liu, Moumita Sinha, Raghavendra Addanki, Aaditya Ramdas, Manas Garg, and Viswanathan Swaminathan. Anytime-valid confidence sequences in an enterprise a/b testing platform. In *Companion Proceedings of the ACM Web Conference 2023*, WWW '23 Companion, page 396–400. Association for Computing Machinery, 2023.

Emilie Kaufmann, Olivier Cappé, and Aurélien Garivier. On the complexity of best-arm identification in multi-armed bandit models. *Journal of Machine Learning Research*, 17(1), 2016. ISSN 1532-4435.

Shengjia Zhao, Enze Zhou, Ashish Sabharwal, and Stefano Ermon. Adaptive concentration inequalities for sequential decision problems. In *Advances in Neural Information Processing Systems*, volume 29, 2016.

D. A. Darling and Herbert Robbins. Some nonparametric sequential tests with power one. *Proceedings of the National Academy of Sciences of the United States of America*, 61(3):804–809, 1968. ISSN 00278424.

P. Massart. The Tight Constant in the Dvoretzky-Kiefer-Wolfowitz Inequality. *The Annals of Probability*, 18(3):1269 – 1283, 1990b.

Seonho Park and Pascal Van Hentenryck. Self-supervised primal-dual learning for constrained optimization, 2022. URL https://arxiv.org/abs/2208.09046.

Zhichao Jia and Benjamin Grimmer. First-order methods for nonsmooth nonconvex functional constrained optimization with or without slater points, 2024. URL https://arxiv.org/abs/2212.00927.

É. A. Nadaraya. On non-parametric estimates of density functions and regression curves. *Theory of Probability & Its Applications*, 10(1):186–190, 1965. doi: 10.1137/1110024. URL https://doi.org/10.1137/1110024.

Bolei Zhou, Agata Lapedriza, Aditya Khosla, Aude Oliva, and Antonio Torralba. Places: A 10 million image database for scene recognition. *IEEE transactions on pattern analysis and machine intelligence*, 40(6):1452–1464, 2017.

Li Deng. The mnist database of handwritten digit images for machine learning research. *IEEE Signal Processing Magazine*, 29(6):141–142, 2012.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale, 2021. URL https://arxiv.org/abs/2010.11929.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition, 2015. URL https://arxiv.org/abs/1512.03385.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2017. URL https://arxiv.org/abs/1412.6980.

Terrance DeVries and Graham W. Taylor. Learning confidence for out-of-distribution detection in neural networks, 2018. URL https://arxiv.org/abs/1802.04865.

Guangyao Chen, Peixi Peng, Xiangqian Wang, and Yonghong Tian. Adversarial reciprocal points learning for open set recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, page 1–1, 2021. ISSN 1939-3539. doi: 10.1109/tpami.2021.3106743. URL http://dx.doi.org/10.1109/TPAMI.2021.3106743.

Rui Huang and Yixuan Li. Mos: Towards scaling out-of-distribution detection for large semantic space, 2021. URL https://arxiv.org/abs/2105.01879.

Hongxin Wei, Renchunzi Xie, Hao Cheng, Lei Feng, Bo An, and Yixuan Li. Mitigating neural network overconfidence with logit normalization, 2022c. URL https://arxiv.org/abs/2205.09310.

Julian Katz-Samuels, Julia Nakhleh, Robert Nowak, and Yixuan Li. Training ood detectors in their natural habitats, 2022b. URL https://arxiv.org/abs/2202.03299.

David Stutz, Krishnamurthy Dj Dvijotham, Ali Taylan Cemgil, and Arnaud Doucet. Learning optimal conformal classifiers. In *International Conference on Learning Representations*, 2022.

Reinhard Heckel, Nihar B Shah, Kannan Ramchandran, and Martin J Wainwright. Active ranking from pairwise comparisons and when parametric assumptions do not help. *The Annals of Statistics*, 47(6):3099–3126, 2019.

Ramya Korlakai Vinayak. *Graph Clustering: Algorithms, analysis and query design*. California Institute of Technology, 2018.

Nagarajan Natarajan, Inderjit S Dhillon, Pradeep K Ravikumar, and Ambuj Tewari. Learning with noisy labels. *Advances in neural information processing systems*, 26, 2013b.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023.

Databricks. Introducing DBRX: A New State-of-the-Art Open LLM, 2024.

Marah Abdin, Sam Ade Jacobs, Ammar Ahmad Awan, Jyoti Aneja, Ahmed Awadallah, Hany Awadalla, Nguyen Bach, Amit Bahree, Arash Bakhtiari, Harkirat Behl, et al. Phi-3 technical report: A highly capable language model locally on your phone. *arXiv preprint arXiv:2404.14219*, 2024.

Qiaoyu Tang, Ziliang Deng, Hongyu Lin, Xianpei Han, Qiao Liang, Boxi Cao, and Le Sun. Toolalpaca: Generalized tool learning for language models with 3000 simulated cases. *arXiv preprint arXiv:2306.05301*, 2023.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding. In *International Conference on Learning Representations*, 2021.

Lea Krause, Wondimagegnhue Tufa, Selene Baez Santamaria, Angel Daza, Urja Khurana, and Piek Vossen. Confidently wrong: Exploring the calibration and expression of (un)certainty of large language models in a multilingual setting. In *Proceedings of the Workshop on Multimodal, Multilingual Natural Language Generation and Multilingual WebNLG Challenge (MM-NLG 2023)*. Association for Computational Linguistics, 2023.

Tobias Groot and Matias Valdenegro Toro. Overconfidence is key: Verbalized uncertainty evaluation in large language and vision-language models. In *Proceedings of the 4th Workshop on Trustworthy Natural Language Processing (TrustNLP 2024)*. Association for Computational Linguistics, 2024.

Chujie Zheng, Hao Zhou, Fandong Meng, Jie Zhou, and Minlie Huang. Large language models are not robust multiple choice selectors. In *The Twelfth International Conference on Learning Representations*, 2024.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

Anastasios N. Angelopoulos, Stephen Bates, Emmanuel J. Candès, Michael I. Jordan, and Lihua Lei. Learn then test: Calibrating predictive algorithms to achieve risk control. *arXiv preprint arXiv:2110.01052*, 2022.

Jase Clarkson, Wenkai Xu, Mihai Cucuringu, and Gesine Reinert. Split conformal prediction under data contamination. *arXiv preprint arXiv:2407.07700*, 2024.

Harris Papadopoulos, Kostas Proedrou, Volodya Vovk, and Alex Gammerman. Inductive confidence machines for regression. In *Machine learning: ECML 2002: 13th European conference on machine learning Helsinki, Finland, August 19–23, 2002 proceedings 13*. Springer, 2002.

Jing Lei, Max G'Sell, Alessandro Rinaldo, Ryan J Tibshirani, and Larry Wasserman. Distribution-free predictive inference for regression. *Journal of the American Statistical Association*, 113 (523), 2018.

Stephanie Lin, Jacob Hilton, and Owain Evans. TruthfulQA: Measuring how models mimic human falsehoods. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, 2022.

Yu Meng, Mengzhou Xia, and Danqi Chen. SimPO: Simple preference optimization with a reference-free reward. *arXiv preprint arXiv:2405.14734*, 2024.

Allen Z. Ren, Anushri Dixit, Alexandra Bodrova, Sumeet Singh, Stephen Tu, Noah Brown, Peng Xu, Leila Takayama, Fei Xia, Jake Varley, Zhenjia Xu, Dorsa Sadigh, Andy Zeng, and Anirudha Majumdar. Robots that ask for help: Uncertainty alignment for large language model planners. In *7th Annual Conference on Robot Learning*, 2023b.

Fanghua Ye, Mingming Yang, Jianhui Pang, Longyue Wang, Derek F. Wong, Emine Yilmaz, Shuming Shi, and Zhaopeng Tu. Benchmarking llms via uncertainty quantification. *arXiv preprint arXiv:2401.12794*, 2024.

Victor Quach, Adam Fisch, Tal Schuster, Adam Yala, Jae Ho Sohn, Tommi S. Jaakkola, and Regina Barzilay. Conformal language modeling. In *The Twelfth International Conference on Learning Representations*, 2024.

Nicolas Deutschmann, Marvin Alberts, and María Rodríguez Martínez. Conformal autoregressive generation: Beam search with coverage guarantees. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2024.

Shauli Ravfogel, Yoav Goldberg, and Jacob Goldberger. Conformal nucleus sampling. In *Findings of the Association for Computational Linguistics: ACL 2023*.

Dennis Ulmer, Chrysoula Zerva, and Andre Martins. Non-exchangeable conformal language generation with nearest neighbors. In *Findings of the Association for Computational Linguistics: EACL 2024*, 2024.

Yu Bai, Song Mei, Huan Wang, Yingbo Zhou, and Caiming Xiong. Efficient and differentiable conformal prediction with general function classes. In *The Tenth International Conference on Learning Representations*, 2022.

Yachong Yang and Arun Kumar Kuchibhotla. Selection and Aggregation of Conformal Prediction Sets. *Journal of the American Statistical Association*, 2024.

Ran Xie, Rina Foygel Barber, and Emmanuel J. Candès. Boosted Conformal Prediction Intervals. *arXiv preprint arXiv:2406.07449*, 2024.

Shayan Kiyani, George Pappas, and Hamed Hassani. Length Optimization in Conformal Prediction. *arXiv preprint arXiv:2406.18814*, 2024.

Yingfan Wang, Haiyang Huang, Cynthia Rudin, and Yaron Shaposhnik. Understanding how dimension reduction tools work: An empirical approach to deciphering t-sne, umap, trimap, and pacmap for data visualization. *Journal of Machine Learning Research*, 22(201):1–73, 2021c.

Michel Ledoux and Michel Talagrand. Probability in banach spaces. 1991.

Mehryar Mohri, Afshin Rostamizadeh, and Ameet Talwalkar. *Foundations of Machine Learning*. The MIT Press, 2012. ISBN 026201825X.

PyTorch. Cifar-10 pytorch tutorial. https://pytorch.org/tutorials/beginner/blitz/cifar10_tutorial.html, 2022. Accessed: 2022-11-18.

Yann LeCun, Sumit Chopra, Raia Hadsell, M Ranzato, and Fujie Huang. A tutorial on energy-based learning. *Predicting structured data*, 1(0), 2006.

Stephen Hanson and Lorien Pratt. Comparing biases for minimal network construction with back-propagation. In D. Touretzky, editor, *Advances in Neural Information Processing Systems*, volume 1. Morgan-Kaufmann, 1988.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

Richard Baraniuk, Mark A Davenport, Marco F Duarte, and Chinmay Hegde. An introduction to compressive sensing. 2014.

James Demmel. The component-wise distance to the nearest singular matrix. *SIAM Journal on Matrix Analysis and Applications*, 13(1):10–19, 1992. doi: 10.1137/0613003.

Hongyi Zhang and Suvrit Sra. First-order methods for geodesically convex optimization. In *Conference on Learning Theory, COLT 2016*, 2016.

R.B. Bapat and Sivaramakrishnan Sivasubramanian. Squared distance matrix of a tree: Inverse and inertia. *Linear Algebra and its Applications*, 491:328–342, 2016.

Andrey Kolmogorov. Über das gesetz des iterierten logarithmus. *Mathematische Annalen*, 101: 126–135, 1929.

Nikolai Smirnov. Approximate laws of distribution of random variables from empirical data. *Uspekhi Matematicheskikh Nauk*, 10:179–206, 1944.

Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *International conference on machine learning*, pages 1597–1607. PMLR, 2020.

Jinsong Zhang, Qiang Fu, Xu Chen, Lun Du, Zelin Li, Gang Wang, xiaoguang Liu, Shi Han, and Dongmei Zhang. Out-of-distribution detection based on in-distribution data patterns memorization with modern hopfield energy. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=KkazG4lgKL.

David Dohan, Winnie Xu, Aitor Lewkowycz, Jacob Austin, David Bieber, Raphael Gontijo Lopes, Yuhuai Wu, Henryk Michalewski, Rif A. Saurous, Jascha Sohl-dickstein, Kevin Murphy, and Charles Sutton. Language model cascades, 2022.

Neha Gupta, Harikrishna Narasimhan, Wittawat Jitkrittum, Ankit Singh Rawat, Aditya Krishna Menon, and Sanjiv Kumar. Language model cascades: Token-level uncertainty and beyond. *arXiv preprint arXiv:2404.10136*, 2024b.

Dharmesh Tailor, Aditya Patra, Rajeev Verma, Putra Manggala, and Eric Nalisnick. Learning to defer to a population: A meta-learning approach. In *International Conference on Artificial Intelligence and Statistics*, pages 3475–3483. PMLR, 2024b.

Adam Fisch, Tommi S. Jaakkola, and Regina Barzilay. Calibrated selective classification. *Transactions on Machine Learning Research*, 2022. ISSN 2835-8856.

Appendix A

Appendix to Chapter 3

The Appendix is organized as follows. We give details of the definitions and notation in Section A.1. The notations are also summarized in the Table A.1. Then we give the proof of the main theorem (Theorem 3.3.1), followed by proofs of supporting lemmas in section A.2. We provide details of its instantiation for finite VC-dimension hypothesis classes and the homogeneous linear separators case in Section A.2.3. Then, we provide the technical details of the lower bound (Lemma 3.3.2). Then we provide details of additional experiments in Section A.3. In Section A.3.4, we provide insights into auto-labeling using PaCMAP (Wang et al., 2021c) visualizations of auto-labeled regions in each round.

A.1 Definitions and Notation

A.1.1 Basic Definitions

Let $\mathcal{X}$ be the instance space and $p(\mathbf{x})$ be a density function supported on $\mathcal{X}$. For any $\mathbf{x}_i \in \mathcal{X}$ let y_i be its true label. Let $X = \{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$ be a set of N i.i.d samples drawn from $\mathcal{X}$. Let set $\mathcal{S} \subseteq \mathcal{X}$ denote a non-empty sub-region of $\mathcal{X}$ and $\mathrm{S}' \subseteq X \cap \mathcal{S}$ be a set of $n > 0$ i.i.d. samples.

Definition A.1.1. (Hypothesis Class with Abstain) We can think of the function g along with set T as inducing an extended hypothesis class $\mathcal{H}^{(T,g)}$. Let $\mathcal{H}^{T,g} = \mathcal{H} \times T$. For any function

$(h,t) \in \mathcal{H}^{(T,g)}$ is defined as

$$(h,t)(\mathbf{x}) := \begin{cases} h(\mathbf{x}) & \text{if } g(h,\mathbf{x}) \geq t \\ \\ \perp & \text{o.w.} \end{cases} \tag{A.1}$$

Here $(h,t)(\mathbf{x}) = \perp$ means the hypothesis (h,t) abstains in classifying the point $\mathbf{x}$. Otherwise, it is equal to $h(\mathbf{x})$.

The subset $\mathcal{S}(h,t) \subseteq \mathcal{S}$ where (h,t) does not abstain and its complement $\bar{\mathcal{S}}(h,t)$ where (h,t) abstains, are defined as follows,

$$\mathcal{S}(h,t) := \{\mathbf{x} \in \mathcal{S} : (h,t)(\mathbf{x}) \neq \perp\}, \qquad \bar{\mathcal{S}}(h,t) := \{\mathbf{x} \in \mathcal{S} : (h,t)(\mathbf{x}) = \perp\}$$

Probability Definitions. The probability $\mathbb{P}(\mathcal{S})$ of subset $\mathcal{S} \subseteq \mathcal{X}$ and the conditional probability of any subset $\mathcal{S}' \subseteq \mathcal{S}$ are given as follows,

$$\mathbb{P}(\mathcal{S}) := \mathbb{P}(\mathcal{S} \mid \mathcal{X}) := \int_{\mathbf{x} \in \mathcal{S}} p(\mathbf{x}) d\mathbf{x}, \qquad \mathbb{P}(\mathcal{S}' \mid \mathcal{S}) := \frac{\mathbb{P}(\mathcal{S}' \mid \mathcal{X})}{\mathbb{P}(\mathcal{S} \mid \mathcal{X})}, \qquad \mathbb{P}(h,t \mid \mathcal{S}) := \mathbb{P}(\mathcal{S}(h,t) \mid \mathcal{S})$$

The empirical probabilities of S' and $\mathrm{S}'' \subseteq S$ are defined as follows,

$$\widehat{\mathbb{P}}(\mathrm{S}') := \frac{|\mathrm{S}'|}{|X|}, \qquad \widehat{\mathbb{P}}(\mathrm{S}''|\mathrm{S}') := \frac{|\mathrm{S}''|}{|\mathrm{S}'|}, \qquad \widehat{\mathbb{P}}(h,t \mid \mathrm{S}') := \frac{|\mathrm{S}'(h,t)|}{|\mathrm{S}'|}$$

Loss Functions. The loss functions are defined as follows,

$$\ell_{0-1}(h,\mathbf{x},y) := \mathbb{1}(h(\mathbf{x}) \neq y),$$

$$\ell_{\perp}(h,t,\mathbf{x}) := \mathbb{1}(g(h,\mathbf{x}) \geq t),$$

$$\ell_{0-1}^{\perp}(h,t,\mathbf{x},y) := \ell_{0-1}(h,\mathbf{x},y) \cdot \ell_{\perp}(h,t,\mathbf{x}).$$

Error Definitions. Define the conditional error in set $\mathcal{S} \subseteq \mathcal{X}$ as follows,

$$\mathcal{E}(h, t \mid \mathcal{S}) := \mathbb{E}_{\mathbf{x}|\mathcal{S}}[\ell_{0-1}^{\perp}(h, t, \mathbf{x}, y)] = \int_{\mathbf{x} \in \mathcal{S}} \ell_{0-1}^{\perp}(h, t, \mathbf{x}, y) \cdot \frac{p(\mathbf{x})}{\mathbb{P}(\mathcal{S})} d\mathbf{x}$$

Then, the conditional error in set $\mathcal{S}(h, t)$ i.e. the subset of $\mathcal{S}$ on which (h, t) does not abstain,

$$\mathcal{E}_a(h, t \mid \mathcal{S}) := \mathcal{E}(h, t \mid \mathcal{S}(h, t)) := \mathbb{E}_{\mathbf{x}|\mathcal{S}(h,t)}[\ell_{0-1}^{\perp}(h, t, \mathbf{x}, y)] = \frac{\mathcal{E}(h, t \mid \mathcal{S})}{\mathbb{P}(h, t \mid \mathcal{S})}$$

Similarly, define their empirical counterparts as follows,

$$\widehat{\mathcal{E}}(h, t \mid \mathrm{S}') := \frac{1}{|\mathrm{S}'|} \sum_{\mathbf{x}_i \in \mathrm{S}'} \ell_{0-1}^{\perp}(h, t, \mathbf{x}_i, y_i),$$
$$\widehat{\mathcal{E}}_a(h, t \mid \mathrm{S}') := \widehat{\mathcal{E}}(h, t \mid \mathrm{S}'(h, t)) := \frac{1}{|\mathrm{S}'(h, t)|} \sum_{\mathbf{x}_i \in \mathrm{S}'(h,t)} \ell_{0-1}^{\perp}(h, t, \mathbf{x}_i, y_i),$$

Note that,

$$\sum_{\mathbf{x}_i \in S} \ell_{0-1}^{\perp}(h, t, \mathbf{x}_i, y_i) = \sum_{\mathbf{x}_i \in S(h,t)} \ell_{0-1}^{\perp}(h, t, \mathbf{x}_i, y_i) = \sum_{\mathbf{x}_i \in S(h,t)} \ell_{0-1}(h, \mathbf{x}_i, y_i)$$

Rademacher Complexity. The Rademacher complexities for the function classes induced by the $\mathcal{H}, T, g$ and the loss functions are defined as follows,

$$\mathfrak{R}_n\big(\mathcal{H}, \ell_{0-1}\big) := \mathbb{E}_{\sigma, \mathrm{S}'}\Big[\sup_{h \in \mathcal{H}} \frac{1}{n} \sum_{i=1}^{n} \sigma_i \ell_{0-1}(h, \mathbf{x}_i, y_i)\Big].$$
$$\mathfrak{R}_n\big(\mathcal{H}^{T,g}, \ell_{\perp}\big) := \mathbb{E}_{\sigma, \mathrm{S}'}\Big[\sup_{(h,t) \in \mathcal{H}^{T,g}} \frac{1}{n} \sum_{i=1}^{n} \sigma_i \ell_{\perp}(h, t, \mathbf{x}_i)\Big].$$
$$\mathfrak{R}_n\big(\mathcal{H}^{T,g}, \ell_{0-1}^{\perp}\big) := \mathbb{E}_{\sigma, \mathrm{S}'}\Big[\sup_{(h,t) \in \mathcal{H}^{T,g}} \frac{1}{n} \sum_{i=1}^{n} \sigma_i \ell_{0-1}^{\perp}(h, t, \mathbf{x}_i, y_i)\Big].$$
$$\mathfrak{R}_n\big(\mathcal{H}^{T,g}\big) := \mathfrak{R}_n\big(\mathcal{H}, \ell_{0-1}\big) + \mathfrak{R}_n\big(\mathcal{H}^{T,g}, \ell_{\perp}\big)$$

A.1.2 Glossary

The notation is summarized in Table A.1 below. A more detailed notation is in section A.1.

A.2 Proofs

A.2.1 Proofs for the General Setup

We begin by restating the theorem here and then give the proof.

Theorem 3.3.1. *(Overall Auto-Labeling Error and Coverage) Let k denote the number of rounds of the TBAL Algorithm 1. Let $n_v^{(i)}, n_a^{(i)}$ denote the number of validation and auto-labeled points at epoch i and $n^{(i)} = |X^{(i)}|$. Let $X_{pool}(A_k)$ be the set of auto-labeled points at the end of round k. $N_a^{(k)} = \sum_{i=1}^{k} n_a^{(i)}$ denote the total number of auto-labeled points. Then, for any $\delta \in (0,1)$, with probability at least $1-\delta$,*

$$\widehat{\mathcal{E}}\Big(X_{pool}(A_k)\Big) \leq \sum_{i=1}^{k} \frac{n_a^{(i)}}{N_a^{(k)}} \Bigg(\underbrace{\widehat{\mathcal{E}}_a\Big(\hat{h}_i, \hat{t}_i \mid X_v^{(i)}\Big)}_{(a)} + \frac{4}{p_0} \underbrace{\Big(\mathfrak{R}_{n_v^{(i)}}\big(\mathcal{H}^{T,g}\big) + \frac{2}{p_0}\sqrt{\frac{1}{n_v^{(i)}} \log(\frac{8k}{\delta})} \Big)}_{(b)} \Bigg)$$
$$+ \frac{4}{p_0} \underbrace{\Bigg(\sum_{i=1}^{k} \frac{n_a^{(i)}}{N_a^{(k)}} \mathfrak{R}_{n_a^{(i)}}\big(\mathcal{H}^{T,g}\big) + \sqrt{\frac{k}{N_a^{(k)}} \log(\frac{8k}{\delta})} \Bigg)}_{(c)}, \quad \textit{and}$$

$$\widehat{\mathcal{P}}(X_{pool}(A_k)) \geq \sum_{i=1}^{k} \mathbb{P}\big(\mathcal{X}^{(i)}(\hat{h}_i, \hat{t}_i)\big) - 2\mathfrak{R}_{n^{(i)}}\big(\mathcal{H}^{T,g}\big) - \sqrt{\frac{2k^2}{N} \log\Big(\frac{8k}{\delta}\Big)}.$$

Proof. Recall the definition of auto-labeling error,

$$\widehat{\mathcal{E}}\Big(X_{pool}(A_k)\Big) = \sum_{i=1}^{k} \frac{m_a^{(i)}}{N_a^{(k)}}, \qquad m_a^{(i)} = n_a^{(i)} \cdot \widehat{\mathcal{E}}_a\big(\hat{h}_i, \hat{t}_i \mid X^{(i)}\big).$$

Here, $m_a^{(i)}$ is the number of auto-labeling mistakes made by the Algorithm in the ith round and

$\widehat{\mathcal{E}}_a\big(\hat{h}_i, \hat{t}_i \mid X^{(i)}\big)$ is the auto-labeling error in that round. Note that we cannot observe these quantities since the true labels for the auto-labeled points are not available. To estimate the auto-labeling error of each round we make use of validation data. Using the validation data we first get an upper bound on the true error rate of the auto-labeling region i.e. $\mathcal{E}_a\big(\hat{h}_i, \hat{t}_i \mid \mathcal{X}^{(i)}\big)$ in terms of the auto-labeling error on the validation data $\widehat{\mathcal{E}}_a\big(\hat{h}_i, \hat{t}_i \mid X_v^{(i)}\big)$ and then get an upper bound on empirical auto-labeling error rate $\widehat{\mathcal{E}}_a\big(\hat{h}_i, \hat{t}_i \mid X^{(i)}\big)$ using the true error rate of the auto-labeling region.

We get these bounds by application of Lemma A.2.1 with $\delta_3 = \delta/4k$ for each round and then apply union bound over all k epochs. Note that we have to apply the lemma twice, first to get the concentration bound w.r.t the validation data and second to get the concentration w.r.t to the auto-labeled points.

$$\mathcal{E}_a\big(\hat{h}_i, \hat{t}_i \mid \mathcal{X}^{(i)}\big) \leq \widehat{\mathcal{E}}_a\big(\hat{h}_i, \hat{t}_i \mid X_v^{(i)}\big) + \frac{4}{p_0}\mathfrak{R}_{n_v^{(i)}}\big(\mathcal{H}^{T,g}\big) + \frac{2}{p_0}\sqrt{\frac{1}{n_v^{(i)}}\log\Big(\frac{8k}{\delta}\Big)}.$$

$$\widehat{\mathcal{E}}_a\big(\hat{h}_i, \hat{t}_i \mid X^{(i)}\big) \leq \mathcal{E}_a\big(\hat{h}_i, \hat{t}_i \mid \mathcal{X}^{(i)}\big) + \frac{4}{p_0}\mathfrak{R}_{n_a^{(i)}}\big(\mathcal{H}^{T,g}\big) + \frac{2}{p_0}\sqrt{\frac{1}{n_a^{(i)}}\log\Big(\frac{8k}{\delta}\Big)}.$$

Substituting $\mathcal{E}_a\big(\hat{h}_i, \hat{t}_i \mid \mathcal{X}^{(i)}\big)$ by its upper confidence bound on the validation data.

$$\widehat{\mathcal{E}}_a\big(\hat{h}_i, \hat{t}_i \mid X^{(i)}\big) \leq \widehat{\mathcal{E}}_a\big(\hat{h}_i, \hat{t}_i \mid X_v^{(i)}\big) + \frac{2}{p_0}\mathfrak{R}_{n_v^{(i)}}\big(\mathcal{H}^{T,g}\big) + \frac{2}{p_0}\mathfrak{R}_{n_a^{(i)}}\big(\mathcal{H}^{T,g}\big) + \frac{2}{p_0}\sqrt{\frac{1}{n_v^{(i)}}\log\Big(\frac{8k}{\delta}\Big)} + \frac{2}{p_0}\sqrt{\frac{1}{n_a^{(i)}}\log\Big(\frac{8k}{\delta}\Big)}.$$

Having an upper bound on the empirical auto-labeling error for i^{th} round gives us an upper bound on the number of auto-labeling mistakes $m_a^{(i)}$ made in that round. It allows us to upper bound the total auto-labeling mistakes in all k rounds and thus the overall auto-labeling error as detailed below,

$$\widehat{\mathcal{E}}\Big(X_{pool}(A_k)\Big) = \sum_{i=1}^{k} \frac{m_a^{(i)}}{N_a^{(k)}}, \qquad m_a^{(i)} = n_a^{(i)} \cdot \widehat{\mathcal{E}}_a\big(\hat{h}_i, \hat{t}_i \mid X^{(i)}\big).$$

Since we have an upper bound on the empirical auto-labeling error in each round, we have an upper bound for each $m_a^{(i)}$, which are used as follows to get the bound on the auto-labeling error,

$$\begin{aligned}
\widehat{\mathcal{E}}(X_{pool}(A_k)) &= \sum_{i=1}^{k} \frac{m_a^{(i)}}{N_a^{(k)}} \\
&= \sum_{i=1}^{k} \frac{n_a^{(i)}}{N_a^{(k)}} \cdot \frac{m_a^{(i)}}{n_a^{(i)}} \\
&= \sum_{i=1}^{k} \frac{n_a^{(i)}}{N_a^{(k)}} \cdot \widehat{\mathcal{E}}_a\Big(\hat{h}_i, \hat{t}_i \mid X^{(i)}\Big) \\
&\le \sum_{i=1}^{k} \frac{n_a^{(i)}}{N_a^{(k)}} \cdot \left(\mathcal{E}_a\big(\hat{h}_i, \hat{t}_i \mid \mathcal{X}^{(i)}\big) + \frac{4}{p_0}\mathfrak{R}_{n_a^{(i)}}\big(\mathcal{H}^{T,g}\big) + \frac{2}{p_0}\sqrt{\frac{1}{n_a^{(i)}}\log\Big(\frac{8k}{\delta}\Big)}\right) \\
&\le \sum_{i=1}^{k} \frac{n_a^{(i)}}{N_a^{(k)}} \cdot \left(\widehat{\mathcal{E}}_a\big(\hat{h}_i, \hat{t}_i \mid X_v^{(i)}\big) + \frac{4}{p_0}\mathfrak{R}_{n_v^{(i)}}\big(\mathcal{H}^{T,g}\big) + \frac{2}{p_0}\sqrt{\frac{1}{n_v^{(i)}}\log\Big(\frac{8k}{\delta}\Big)}\right. \\
&\qquad\qquad \left. + \frac{4}{p_0}\mathfrak{R}_{n_a^{(i)}}\big(\mathcal{H}^{T,g}\big) + \frac{2}{p_0}\sqrt{\frac{1}{n_a^{(i)}}\log\Big(\frac{8k}{\delta}\Big)}\right) \\
&\le \sum_{i=1}^{k} \frac{n_a^{(i)}}{N_a^{(k)}} \cdot \left(\widehat{\mathcal{E}}_a\big(\hat{h}_i, \hat{t}_i \mid X_v^{(i)}\big) + \frac{4}{p_0}\mathfrak{R}_{n_v^{(i)}}\big(\mathcal{H}^{T,g}\big) + \frac{4}{p_0}\mathfrak{R}_{n_a^{(i)}}\big(\mathcal{H}^{T,g}\big) + \right. \\
&\qquad\qquad \left. \frac{2}{p_0}\sqrt{\frac{1}{n_v^{(i)}}\log\Big(\frac{8k}{\delta}\Big)}\right) + \sum_{i=1}^{k} \frac{n_a^{(i)}}{N_a^{(k)}} \cdot \left(\frac{2}{p_0}\sqrt{\frac{1}{n_a^{(i)}}\log\Big(\frac{8k}{\delta}\Big)}\right)
\end{aligned}$$

The last term is simplified as follows,

$$\begin{aligned}
\sum_{i=1}^{k} \frac{n_a^{(i)}}{N_a^{(k)}} \cdot \left(\frac{2}{p_0}\sqrt{\frac{1}{n_a^{(i)}}\log\Big(\frac{8k}{\delta}\Big)}\right) &= \frac{2}{p_0} \cdot \sum_{i=1}^{k} \frac{n_a^{(i)}}{N_a^{(k)}}\sqrt{\frac{1}{n_a^{(i)}}\log\Big(\frac{8k}{\delta}\Big)} \\
&= \frac{2}{p_0} \cdot \frac{1}{N_a^{(k)}} \sum_{i=1}^{k} \sqrt{n_a^{(i)}\log\Big(\frac{8k}{\delta}\Big)} \\
&= \frac{2}{p_0} \cdot \sqrt{\log\Big(\frac{8k}{\delta}\Big)} \cdot \sum_{i=1}^{k} \frac{\sqrt{n_a^{(i)}}}{N_a^{(k)}}
\end{aligned}$$

$$\leq \frac{2}{p_0} \cdot \sqrt{\log\left(\frac{8k}{\delta}\right)} \cdot \sqrt{\frac{k}{N_a^{(k)}}}$$
$$= \frac{2}{p_0} \cdot \sqrt{\frac{k}{N_a^{(k)}} \log\left(\frac{8k}{\delta}\right)}$$

The last inequality follows from the application of the inequality $||\mathbf{u}||_1 \leq \sqrt{k}||\mathbf{u}||_2$ for any vector $\mathbf{u} \in \mathbb{R}^k$. Here we let $\mathbf{u} = [\sqrt{n_a}^{(1)}, \ldots, \sqrt{n_a}^{(k)}]$, and since $\forall i \ \sqrt{n_a^{(i)}} > 0$ so, $\sum_{i=1}^k \sqrt{n_a}^{(i)} = ||\mathbf{u}||_1$ and $N_a^{(k)} = ||\mathbf{u}||_2^2$.

$$\frac{\sum_{i=1}^k \sqrt{n_a^{(i)}}}{N_a^{(k)}} = \frac{||\mathbf{u}||_1}{||\mathbf{u}||_2^2} \leq \frac{\sqrt{k}||\mathbf{u}||_2}{||\mathbf{u}||_2^2} = \frac{\sqrt{k}}{||\mathbf{u}||_2} = \sqrt{\frac{k}{N_a^{(k)}}}$$

To get the bound on coverage we follow the same steps except that we can use all the unlabeled pool of size $n^(i)$ to estimate the coverage in each round which gives us the bound in terms of $n^{(i)}$ and N as follows,

$$\begin{aligned}
\widehat{\varphi}(X_{pool}(A_k)) &= \frac{1}{N} \sum_{i=1}^k n_a^{(i)} \\
&= \frac{1}{N} \sum_{i=1}^k n^{(i)} \cdot \frac{n_a^{(i)}}{n^{(i)}} \\
&= \frac{1}{N} \sum_{i=1}^k n^{(i)} \cdot \widehat{\mathbb{P}}(X_a^{(i)} \mid X^{(i)}) \\
&= \frac{1}{N} \sum_{i=1}^k n^{(i)} \cdot \widehat{\mathbb{P}}(\hat{h}_i, \hat{t}_i \mid X^{(i)}) \\
&\geq \sum_{i=1}^k \frac{n^{(i)}}{N} \left(\mathbb{P}\big(\hat{h}_i, \hat{t}_i \mid \mathcal{X}^{(i)}\big) - 2\mathfrak{R}_{n^{(i)}}\big(\mathcal{H}^{T,g}\big) - \sqrt{\frac{1}{n^{(i)}} \log\left(\frac{8k}{\delta}\right)} \right) \\
&\geq \sum_{i=1}^k \frac{n^{(i)}}{N} \left(\mathbb{P}\big(\hat{h}_i, \hat{t}_i \mid \mathcal{X}^{(i)}\big) - 2\mathfrak{R}_{n^{(i)}}\big(\mathcal{H}^{T,g}\big) \right) - \sqrt{\frac{k}{N} \log\left(\frac{8k}{\delta}\right)}
\end{aligned}$$

We bound the first term as follows,

$$\begin{aligned}\sum_{i=1}^{k} \frac{n^{(i)}}{N} \mathbb{P}\Big(\hat{h}_i, \hat{t}_i \mid \mathcal{X}^{(i)}\Big) &= \sum_{i=1}^{k} \frac{n^{(i)}}{N} \cdot \frac{\mathbb{P}\big(\mathcal{X}^{(i)}(\hat{h}_i, \hat{t}_i)\big)}{\mathbb{P}\big(\mathcal{X}^{(i)}\big)} \\ &\geq \sum_{i=1}^{k} \Big(\mathbb{P}\big(\mathcal{X}^{(i)}\big) - \sqrt{\frac{1}{N}\log\Big(\frac{8k}{\delta}\Big)}\Big) \cdot \Big(\frac{\mathbb{P}\big(\mathcal{X}(\hat{h}_i, \hat{t}_i)\big)}{\mathbb{P}\big(\mathcal{X}^{(i)}\big)}\Big) \\ &\geq \sum_{i=1}^{k} \mathbb{P}\big(\mathcal{X}^{(i)}(\hat{h}_i, \hat{t}_i)\big) - \sqrt{\frac{1}{N}\log\Big(\frac{8k}{\delta}\Big)}\end{aligned}$$

Substituting it back we get,

$$\begin{aligned}\widehat{\mathscr{P}}\big(X_{pool}(A_k)\big) &\geq \sum_{i=1}^{k} \mathbb{P}\big(\mathcal{X}^{(i)}(\hat{h}_i, \hat{t}_i)\big) - 2\mathfrak{R}_{n^{(i)}}\big(\mathscr{H}^{T,g}\big) - k\sqrt{\frac{1}{N}\log\Big(\frac{8k}{\delta}\Big)} - \sqrt{\frac{k}{N}\log\Big(\frac{8k}{\delta}\Big)} \\ &\geq \sum_{i=1}^{k} \mathbb{P}\big(\mathcal{X}^{(i)}(\hat{h}_i, \hat{t}_i)\big) - 2\mathfrak{R}_{n^{(i)}}\big(\mathscr{H}^{T,g}\big) - \sqrt{\frac{4k^2}{N}\log\Big(\frac{8k}{\delta}\Big)}\end{aligned}$$

For the last step we use the inequality $\sqrt{a} + \sqrt{b} \leq \sqrt{2(a+b)}$ for any $a, b \in \mathbb{R}^+$. □

Rademacher complexity and validation error trade-off. The bound contains validation errors at different thresholds. We revisit the definition of validation error appearing in the bound. Let $X_v = \{x_1, \ldots x_{n_v}\}$ be the validation samples and y_i be the label corresponding to x_i. Given any $h \in \mathscr{H}$ and the confidence function g, we have different subsets $X_v(h,t)$ of the validation points for which the model's confidence is higher than t. More precisely, $X_v(h,t) = \{x_i \in X_v : g(h, x_i) \geq t\}$ and the validation error $\hat{\mathcal{E}}_a(h, t \mid X_v)$ is computed on each of these subsets as follows, $\hat{\mathcal{E}}_a(h, t \mid X_v) = \frac{1}{|X_v(h,t)|} \sum_{x_i \in X_v(h,t)} \mathbf{1}(h(x_i) \neq y_i)$ this error is different from the overall validation error which is computed over the entire set of validation points X_v:

$$\hat{\mathcal{E}}(h \mid X_v) = \frac{1}{|X_v|} \sum_{x_i \in X_v} \mathbf{1}(h(x_i) \neq y_i).$$

The TBAL method computes $\hat{\mathcal{E}}_a(h, t \mid X_v)$ at different thresholds (t) and selects the threshold at which it is at most ϵ. Thus even if the overall validation error $\hat{\mathcal{E}}(h \mid X_v)$ is bad, there could

still be regions in the space where the conditional validation error $\hat{\mathcal{E}}_a(h, t \mid X_v)$ is small. This can be easily seen in Figure 3.2 in the paper. Here we are doing auto-labeling using a linear function class that has low Rademacher complexity. All the models in this class have high overall validation error $\hat{\mathcal{E}}(h \mid X_v)$ but there are subsets where the conditional validation error $\hat{\mathcal{E}}_a(h, t \mid X_v)$ is small and TBAL is able to find those subsets. Thus we see that working with low Rademacher complexity classes might lead to high overall validation error. However, it does not affect TBAL as long as there are regions of low conditional validation error $\hat{\mathcal{E}}_a(h, t \mid X_v)$. Furthermore, our upper bound on excess auto-labeling error depends on $\hat{\mathcal{E}}_a(h, t \mid X_v)$ which due to the TBAL procedure is at most ϵ and hence it is not in conflict with the Rademacher complexity term.

Next, we state the result for uniform convergence between $\mathcal{E}_a\big(h, t \mid \mathcal{S}\big), \widehat{\mathcal{E}}_a\big(h, t \mid \mathrm{S}'\big)$ and give its proof.

Lemma A.2.1. *For any $\delta_3, p_0 \in (0, 1)$, let $\mathcal{S}$ and S be defined as above. Let $\mathbb{P}(h, t \mid \mathcal{S}) \geq p_0$ and $\hat{\mathbb{P}}(h, t \mid \mathrm{S}') \geq p_0 \forall (h, t) \in \mathcal{H}^{T,g}$, the following holds w.p. at least $1 - \delta_3/2$*

$$\big|\mathcal{E}_a\big(h, t \mid \mathcal{S}\big) - \widehat{\mathcal{E}}_a\big(h, t \mid \mathrm{S}'\big)\big| \leq \frac{4}{p_0}\mathfrak{R}_n\big(\mathcal{H}^{T,g}\big) + \frac{2}{p_0}\sqrt{\frac{1}{n}\log(\frac{2}{\delta_3})} \qquad \forall (h, t) \in \mathcal{H}^{T,g}. \tag{A.2}$$

Proof. We begin with proving one side of the inequality and the other side is shown by following the same steps. The proof is based on applying the uniform convergence results for $\widehat{\mathcal{E}}(h, t \mid \mathrm{S}')$ and $\widehat{\mathbb{P}}(h, t \mid \mathrm{S}')$ from Lemma A.2.2. The main difficulty here is that $\mathbb{E}_S[\widehat{\mathcal{E}}_a\big(h, t \mid \mathrm{S}'\big)] \neq \mathcal{E}_a\big(h, t \mid \mathcal{S}\big)$, so we cannot directly get the above result from standard uniform convergence bounds.

We prove it, by using the results from the Lemma A.2.2 and restricting the region $\mathcal{S}$ such that it has probability mass at least p_0.

By definitions of $\mathcal{E}_a\big(h, t \mid \mathcal{S}\big)$ and $\widehat{\mathcal{E}}_a\big(h, t \mid \mathrm{S}'\big)$ we have,

$$\mathcal{E}(h, t \mid \mathcal{S}) = \mathbb{P}(h, t \mid \mathcal{S}) \cdot \mathcal{E}_a\big(h, t \mid \mathcal{S}\big) \qquad \text{and} \qquad \widehat{\mathcal{E}}(h, t \mid \mathrm{S}') = \hat{\mathbb{P}}(h, t \mid \mathrm{S}') \cdot \widehat{\mathcal{E}}_a\big(h, t \mid \mathrm{S}'\big).$$

Let $\xi_1 = \sqrt{(1/n)\log(2/\delta_1)}, \xi_2 = \sqrt{(1/n)\log(2/\delta_2)}$. From lemma A.2.2 we have,

$$\mathcal{E}(h,t \mid \mathcal{S}) \leq \widehat{\mathcal{E}}(h,t \mid \mathrm{S}') + 2\mathfrak{R}_n\big(\mathcal{H}^{T,g}\big) + \xi_1 \quad \forall (h,t) \in \mathcal{H}^{T,g} \quad \text{w.p. } 1-\delta_1/2. \tag{A.3}$$

$$\hat{\mathbb{P}}(h,t \mid \mathrm{S}') \leq \mathbb{P}(h,t \mid \mathcal{S}) + 2\mathfrak{R}_n\big(\mathcal{H}^{T,g}\big) + \xi_2 \quad \forall (h,t) \in \mathcal{H}^{T,g} \quad \text{w.p. } 1-\delta_2/2. \tag{A.4}$$

Plugging in the above definitions of errors in equation (A.3) we get,

$$\mathbb{P}(h,t \mid \mathcal{S}) \cdot \mathcal{E}_a\big(h,t \mid \mathcal{S}\big) \leq \widehat{\mathbb{P}}(h,t \mid \mathrm{S}') \cdot \widehat{\mathcal{E}}_a\big(h,t \mid \mathrm{S}'\big) + 2\mathfrak{R}_n\big(\mathcal{H}^{T,g}\big) + \xi_1. \tag{A.5}$$

$$\mathcal{E}_a\big(h,t \mid \mathcal{S}\big) \leq \frac{\widehat{\mathbb{P}}(h,t \mid \mathrm{S}')}{\mathbb{P}(h,t \mid \mathcal{S})}\widehat{\mathcal{E}}_a\big(h,t \mid \mathrm{S}'\big) + 2\frac{\mathfrak{R}_n\big(\mathcal{H}^{T,g}\big)}{\mathbb{P}(h,t \mid \mathcal{S})} + \frac{\xi_1}{\mathbb{P}(h,t \mid \mathcal{S})}. \tag{A.6}$$

Substituting $\widehat{\mathbb{P}}(h,t \mid \mathrm{S}')$ from equation A.4 in the above equation, we get the following w.p. $(1-\delta_1/2)(1-\delta_2/2)$, $\forall (h,t) \in \mathcal{H}^{T,g}$,

$$\begin{aligned}
\mathcal{E}_a\big(h,t \mid \mathcal{S}\big) &\leq \Big(\frac{\mathbb{P}(h,t \mid \mathcal{S}) + 2\mathfrak{R}_n\big(\mathcal{H}^{T,g}\big) + \xi_2}{\mathbb{P}(h,t \mid \mathcal{S})}\Big)\widehat{\mathcal{E}}_a\big(h,t \mid \mathrm{S}'\big) + \frac{2\mathfrak{R}_n\big(\mathcal{H}^{T,g}\big)}{\mathbb{P}(h,t \mid \mathcal{S})} + \frac{\xi_1}{\mathbb{P}(h,t \mid \mathcal{S})} \quad . \\
&= \Big(1 + \frac{2\mathfrak{R}_n\big(\mathcal{H}^{T,g}\big)}{\mathbb{P}(h,t \mid \mathcal{S})} + \frac{\xi_2}{\mathbb{P}(h,t \mid \mathcal{S})}\Big)\widehat{\mathcal{E}}_a\big(h,t \mid \mathrm{S}'\big) + \frac{2\mathfrak{R}_n\big(\mathcal{H}^{T,g}\big)}{\mathbb{P}(h,t \mid \mathcal{S})} + \frac{\xi_1}{\mathbb{P}(h,t \mid \mathcal{S})}. \\
&= \widehat{\mathcal{E}}_a\big(h,t \mid \mathrm{S}'\big) + \frac{2\mathfrak{R}_n\big(\mathcal{H}^{T,g}\big)}{\mathbb{P}(h,t \mid \mathcal{S})} \cdot \widehat{\mathcal{E}}_a\big(h,t \mid \mathrm{S}'\big) + \frac{\xi_2}{\mathbb{P}(h,t \mid \mathcal{S})}\widehat{\mathcal{E}}_a\big(h,t \mid \mathrm{S}'\big) + \frac{2\mathfrak{R}_n\big(\mathcal{H}^{T,g}\big)}{\mathbb{P}(h,t \mid \mathcal{S})} \\
&\qquad + \frac{\xi_1}{\mathbb{P}(h,t \mid \mathcal{S})}.
\end{aligned}$$

Using upper bound $\widehat{\mathcal{E}}_a\big(h,t \mid \mathrm{S}'\big) \leq 1$ in the second and third terms,

$$\mathcal{E}_a\big(h,t \mid \mathcal{S}\big) \leq \widehat{\mathcal{E}}_a\big(h,t \mid \mathrm{S}'\big) + 4\frac{\mathfrak{R}_n\big(\mathcal{H}^{T,g}\big)}{\mathbb{P}(h,t \mid \mathcal{S})} + \frac{\xi_1 + \xi_2}{\mathbb{P}(h,t \mid \mathcal{S})} \qquad \forall (h,t) \in \mathcal{H}^{T,g}\ w.p. \geq 1 - (\delta_1 + \delta_2)/2.$$

Using $\mathbb{P}(h,t \mid \mathcal{S}) \geq p_0$

$$\mathcal{E}_a\big(h,t \mid \mathcal{S}\big) \leq \widehat{\mathcal{E}}_a\big(h,t \mid \mathrm{S}'\big) + \frac{4}{p_0}\mathfrak{R}_n\big(\mathcal{H}^{T,g}\big) + \frac{\xi_1 + \xi_2}{p_0} \qquad \forall (h,t) \in \mathcal{H}^{T,g}\ w.p. \geq 1 - (\delta_1 + \delta_2)/2.$$

Letting $\delta_1 = \delta_2 = \delta_3$ and $\xi_1 = \xi_2 = \frac{\xi \cdot p_0}{2}$ gives $\xi = \sqrt{\frac{4}{p_0^2 n} \log\left(\frac{2}{\delta_3}\right)}$ and

$$\mathcal{E}_a\big(h, t \mid \mathcal{S}\big) \leq \widehat{\mathcal{E}}_a\big(h, t \mid \mathrm{S}'\big) + \frac{4}{p_0}\mathfrak{R}_n\big(\mathcal{H}^{T,g}\big) + \xi \qquad \forall (h,t) \in \mathcal{H}^{T,g} \ w.p. \geq 1 - \delta_3.$$

This proves one side of the result, and the other side of the result follows similarly. □

Lemma A.2.2. *Let $\mathcal{S} \subseteq \mathcal{X}$ be a sub-region of $\mathcal{X}$ and $S = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$ be a set of n i.i.d samples in $\mathcal{S}$ drawn from distribution $P_{\mathbf{x}}$. Let $\{y_1, \ldots y_n\}$ be the corresponding true labels, let $\mathfrak{R}_n\big(\mathcal{H}^{T,g}\big)$ be the rademacher complexity of class $\mathcal{H}^{T,g}$ then for any $\delta_1, \delta_2 \in (0,1)$ we have,*

$$|\mathcal{E}(h, t \mid \mathcal{S}) - \widehat{\mathcal{E}}(h, t \mid \mathrm{S}')| \leq 2\mathfrak{R}_n\big(\mathcal{H}^{T,g}\big) + \sqrt{\frac{1}{n}\log(\frac{2}{\delta_1})} \quad \forall (h,t) \in \mathcal{H}^{T,g} \quad w.p.\ 1 - \delta_1/2. \quad \text{(A.7)}$$

$$|\mathbb{P}(h, t \mid \mathcal{S}) - \hat{\mathbb{P}}(h, t \mid \mathrm{S}')| \leq 2\mathfrak{R}_n\big(\mathcal{H}^{T,g}\big) + \sqrt{\frac{1}{n}\log(\frac{2}{\delta_2})} \quad \forall (h,t) \in \mathcal{H}^{T,g} \quad w.p.\ 1 - \delta_2/2. \quad \text{(A.8)}$$

Proof. The proof is similar to the standard proofs for Rademacher complexity based generalization error bound. Since we work with the modified loss function and hypothesis class to include the abstain option, for completeness, we give the proof here. The proofs for error and probability bounds are very much the same, except for the change in the loss function. We give the proof for the error bound here.

The result follows by applying McDiarmid's inequality on the function $\phi(\mathrm{S}')$ defined as below,

$$\phi(\mathrm{S}') := \sup_{(h,t) \in \mathcal{H}^{T,g}} \mathcal{E}(h, t \mid \mathcal{S}) - \widehat{\mathcal{E}}_S(h, t \mid \mathrm{S}').$$

To apply McDiarmid's inequality, we first show that $\phi(S)$ satisfies the bounded difference property (Lemma A.2.4). This gives us,

$$\mathcal{E}(h,t \mid \mathcal{S}) - \widehat{\mathcal{E}}_S(h,t \mid \mathrm{S}') \leq \phi(\mathrm{S}') \leq \mathbb{E}_{\mathrm{S}'}[\phi(\mathrm{S}')] + \sqrt{\frac{1}{n}\log(\frac{2}{\delta_1})} \quad \forall (h,t) \in \mathcal{H}^{T,g} \quad \text{w.p. } 1 - \frac{\delta_1}{2}.$$

Using the bound on $\mathbb{E}_S[\phi(S)]$ from Lemma A.2.3 we get,

$$\mathcal{E}(h,t \mid \mathcal{S}) \leq \widehat{\mathcal{E}}(h,t \mid \mathrm{S}') + 2\mathfrak{R}_n\big(\mathcal{H}^{T,g}\big) + \sqrt{\frac{1}{n}\log(\frac{2}{\delta_1})} \quad \forall (h,t) \in \mathcal{H}^{T,g} \quad \text{w.p. } 1 - \frac{\delta_1}{2}.$$

Similarly, the bound for the other side is obtained which holds w.p. $1 - \delta_1/2$, and combining both we get eq. (A.7).

The bound of probabilities is obtained by following the same steps as above but with a different loss function, $\ell_\perp$, since $\mathbb{P}(h,t \mid \mathcal{S})$ is the probability mass of the region where (h,t) does not abstain.

□

Lemma A.2.3. *Let $\mathcal{S} \subseteq \mathcal{X}$ be a sub-region of $\mathcal{X}$ and $\mathrm{S}' = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$ be a set of n i.i.d samples in $\mathcal{S}$ drawn from distribution $P_{\mathbf{x}}$. Let $\{y_1, \ldots y_n\}$ be the corresponding true labels and let $\mathfrak{R}_n\big(\mathcal{H}^{T,g}\big)$ be the Rademacher complexity of the function class $\mathcal{H}^{T,g}$ defined over n i.i.d. samples. Then we have,*

$$\mathbb{E}_S\Big[\sup_{(h,t)\in\mathcal{H}^{T,g}} \mathcal{E}(h,t \mid \mathcal{S}) - \widehat{\mathcal{E}}(h,t \mid \mathrm{S}')\Big] \leq 2\mathfrak{R}_n\big(\mathcal{H}^{T,g}\big). \tag{A.9}$$

Proof. Let $\tilde{S} = \{\tilde{\mathbf{x}}_1, \tilde{\mathbf{x}}_2, \ldots \tilde{\mathbf{x}}_n\}$ be another set of independent draws from the same distribution as of S and let the corresponding labels be $\{\tilde{y}_1, \ldots \tilde{y}_n\}$. These samples are usually termed as *ghost samples* and do not need to be counted in the sample complexity.

$$\begin{aligned}
\mathbb{E}_S\Big[\sup_{(h,t)\in\mathcal{H}^{T,g}} \mathcal{E}(h,t \mid \mathcal{S}) - \widehat{\mathcal{E}}(h,t \mid \mathrm{S}')\Big] &= \mathbb{E}_S\Big[\sup_{(h,t)\in\mathcal{H}^{T,g}} \mathbb{E}_{\tilde{S}}\big[\widehat{\mathcal{E}}(h,t \mid \tilde{S})\big] - \widehat{\mathcal{E}}(h,t \mid \mathrm{S}')\Big]. \\
&= \mathbb{E}_S\Big[\sup_{(h,t)\in\mathcal{H}^{T,g}} \mathbb{E}_{\tilde{S}}\big[\widehat{\mathcal{E}}(h,t \mid \tilde{S}) - \widehat{\mathcal{E}}(h,t \mid \mathrm{S}')\big]\Big]. \\
&\leq \mathbb{E}_S\Big[\mathbb{E}_{\tilde{S}}\Big[\sup_{(h,t)\in\mathcal{H}^{T,g}} \big[\widehat{\mathcal{E}}(h,t \mid \tilde{S}) - \widehat{\mathcal{E}}(h,t \mid \mathrm{S}')\big]\Big]\Big].
\end{aligned}$$

$$
\begin{aligned}
&= \mathbb{E}_{S,\tilde{S}}\Big[\sup_{h\in\mathcal{H}^{T,g}}\big[\widehat{\mathcal{E}}(h,t\mid\tilde{S})-\widehat{\mathcal{E}}(h,t\mid \mathrm{S}')\big]\Big].\\
&= \mathbb{E}_{S,\tilde{S}}\Big[\sup_{(h,t)\in\mathcal{H}^{T,g}}\Big[\frac{1}{n}\sum_{i=1}^{n}\ell_{0-1}^{\perp}(h,t,\tilde{\mathbf{x}}_i,\tilde{y}_i)-\frac{1}{n}\sum_{i=1}^{n}\ell_{0-1}^{\perp}(h,t,\mathbf{x}_i,y_i)\Big]\Big].\\
&= \mathbb{E}_{\sigma,S,\tilde{S}}\Big[\sup_{h\in\mathcal{H}^{T,g}}\Big[\frac{1}{n}\sum_{i=1}^{n}\sigma_i\ell_{0-1}^{\perp}(h,t,\tilde{\mathbf{x}}_i,\tilde{y}_i)-\frac{1}{n}\sum_{i=1}^{n}\sigma_i\ell_{0-1}^{\perp}(h,t,\mathbf{x}_i,y_i)\Big]\Big].\\
&\leq \mathbb{E}_{\sigma,\tilde{S}}\Big[\sup_{(h,t)\in\mathcal{H}^{T,g}}\frac{1}{n}\sum_{i=1}^{n}\sigma_i\ell_{0-1}^{\perp}(h,t,\tilde{\mathbf{x}}_i,\tilde{y}_i)\Big]+\\
&\qquad\qquad\qquad\qquad\mathbb{E}_{\sigma,S}\Big[\sup_{(h,t)\in\mathcal{H}^{T,g}}\frac{1}{n}\sum_{i=1}^{n}\sigma_i\ell_{0-1}^{\perp}(h,t,\mathbf{x}_i,y_i)\Big].\\
&= 2\mathfrak{R}_n\big(\mathcal{H}^{T,g},\ell_{0-1}^{\perp}\big).\\
&\leq 2\mathfrak{R}_n\big(\mathcal{H}^{T,g}\big).
\end{aligned}
$$

In the last step, we used the upper bound on the Rademacher complexity from Lemma A.2.5. □

Lemma A.2.4. *(Bounded Difference) Let* S' *be a set of i.i.d samples from* $P_{\mathbf{x}}$ *then for* $\phi(\mathrm{S}') := \sup_{(h,t)\in\mathcal{H}^{T,g}}\mathcal{E}(h,t\mid\mathcal{S})-\widehat{\mathcal{E}}(h,t\mid\mathrm{S}')$*, with probability at least* $1-\delta$*,*

$$
\phi(\mathrm{S}')\leq\mathbb{E}_{\mathrm{S}'}[\phi(\mathrm{S}')]+\sqrt{\frac{1}{|\mathrm{S}'|}\log(\frac{1}{\delta})} \tag{A.10}
$$

Proof. It is proved by showing that $\phi(\mathrm{S}')$ satisfies the conditions (in particular the bounded difference assumption) needed for the application of McDiarmid Inequality. To see this, Let $\mathrm{S}'=\{\mathbf{x}_1,\mathbf{x}_2,\dots\mathbf{x}_i,\dots,\mathbf{x}_n\}$ and let $\mathrm{S}''=\{\mathbf{x}_1,\mathbf{x}_2,\dots\mathbf{x}_i',\dots,\mathbf{x}_n\}$, i.e. S' and S'' may differ only on the i^{th} sample.

$$
\begin{aligned}
|\phi(\mathrm{S}')-\phi(\mathrm{S}'')| &= \Big|\sup_{(h,t)\in\mathcal{H}^{T,g}}\mathcal{E}(h,t\mid\mathcal{S})-\widehat{\mathcal{E}}(h,t\mid\mathrm{S}')-\sup_{(h,t)\in\mathcal{H}^{T,g}}\mathcal{E}(h,t\mid\mathcal{S})-\widehat{\mathcal{E}}(h,t\mid\mathrm{S}'')\Big|.\\
&\leq \Big|\sup_{(h,t)\in\mathcal{H}^{T,g}}\Big(\mathcal{E}(h,t\mid\mathcal{S})-\widehat{\mathcal{E}}(h,t\mid\mathrm{S}')-\mathcal{E}(h,t\mid\mathcal{S})+\widehat{\mathcal{E}}(h,t\mid\mathrm{S}'')\Big)\Big|.\\
&= \Big|\sup_{(h,t)\in\mathcal{H}^{T,g}}\Big(\widehat{\mathcal{E}}(h,t\mid\mathrm{S}')-\widehat{\mathcal{E}}(h,t\mid\mathrm{S}'')\Big)\Big|.\\
&= \Big|\sup_{(h,t)\in\mathcal{H}^{T,g}}\Big(\frac{1}{|\mathrm{S}'|}\sum_{\mathbf{z}_j\in\mathrm{S}'}\ell_{0-1}^{\perp}(h,t,\mathbf{x}_j,y_j)-\frac{1}{|\mathrm{S}''|}\sum_{\mathbf{z}_j\in\mathrm{S}''}\ell_{0-1}^{\perp}(h,t,\mathbf{x}_j,y_j)\Big)\Big|.
\end{aligned}
$$

$$
\begin{aligned}
&= \Big| \sup_{(h,t)\in\mathcal{H}^{T,g}} \Big(\frac{1}{n} \sum_{j\neq i} \big(\ell_{0-1}^{\perp}(h,t,\mathbf{x}_j,y_j) - \ell_{0-1}^{\perp}(h,t,\mathbf{x}_j,y_j)\big) \quad + \\
&\qquad \frac{1}{n}\big(\ell_{0-1}^{\perp}(h,t,\mathbf{x}_i,y_i) - \ell_{0-1}^{\perp}(h,t,\mathbf{x}_i',y_i')\big)\Big)\Big|. \\
&= \Big| \sup_{(h,t)\in\mathcal{H}^{T,g}} \Big(\frac{1}{n}\big(\ell_{0-1}^{\perp}(h,t,\mathbf{x}_i,y_i) - \ell_{0-1}^{\perp}(h,t,\mathbf{x}_i',y_i')\big)\Big)\Big|. \\
&\leq \frac{1}{n}
\end{aligned}
$$

The last step follows since $\ell_{0-1}^{\perp}$ is a 0-1 loss function so letting $\ell_{0-1}^{\perp}(h,t,\mathbf{x}_i,y_i) = 1$ and $\ell_{0-1}^{\perp}(h,t,\mathbf{x}_i',y_i') = 0$ gives an upper bound on the difference. Thus, we can apply McDiarmid's Inequality here and get the bound.

□

The relationship between the Rademacher complexities is obtained using the following Lemma A.2.5 due to (DeSalvo et al., 2015).

Lemma A.2.5. *((DeSalvo et al., 2015)) Let $\ell_{0-1}, \ell_{\perp}, \ell_{0-1}^{\perp}$ be the loss functions defined as above and the Rademacher complexities on n i.i.d. samples S be $\mathfrak{R}_n\big(\mathcal{H}, \ell_{0-1}\big)$, $\mathfrak{R}_n\big(\mathcal{H}^{T,g}, \ell_{\perp}\big)$, $\mathfrak{R}_n\big(\mathcal{H}^{T,g}, \ell_{0-1}^{\perp}\big)$ respectively. Then,*

$$
\mathfrak{R}_n\big(\mathcal{H}^{T,g}, \ell_{0-1}^{\perp}\big) \leq \mathfrak{R}_n\big(\mathcal{H}, \ell_{0-1}\big) + \mathfrak{R}_n\big(\mathcal{H}^{T,g}, \ell_{\perp}\big) =: \mathfrak{R}_n\big(\mathcal{H}^{T,g}\big). \tag{A.11}
$$

Detailed proof of this lemma can be found in (DeSalvo et al., 2015). The result follows by expressing $\ell_{0-1} \cdot \ell_{\perp}$ as $(\ell_{0-1} + \ell_{\perp} - 1)_+$ and then applying Talagrand's contraction lemma (Ledoux and Talagrand, 1991).

A.2.2 Bounds for Finite VC-Dimension Classes

Here we specialize the auto-labeling error and coverage bounds to the setting of finite VC-dimension classes and then instantiate for a specific setting of homogeneous linear classifiers and uniform distribution.

Lemma A.2.6. *(Mohri et al., 2012) (Corollary 3.8 and 3.18). Let the VC-dimension of function class induced by $\mathcal{F}$ be any class of functions from $\mathcal{X} \mapsto \mathcal{Y} \cup \{\perp\}$, and $\ell : \mathcal{Y} \cup \{\perp\} \mapsto \{0, 1\}$ be a 0-1 function. Then,*

$$\mathfrak{R}_n(\mathcal{F}, \ell) \leq \sqrt{\frac{2\mathcal{V}(\mathcal{F}, \ell)}{n} \log\Big(\frac{en}{\mathcal{V}(\mathcal{F}, \ell)}\Big)}. \tag{A.12}$$

Corollary A.2.6.1. *(Auto-Labeling Error and Coverage for Finite VC-dimension Classes) Let k denote the number of rounds of TBAL algorithm 1. Let $\mathcal{V}(\mathcal{H}^{T,g}) = d$ Let $X_{pool}(A_k)$ be the set of auto-labeled points at the end of round k. $N_a^{(k)} = \sum_{i=1}^{k} n_a^{(i)}$ denote the total number of auto-labeled points. With probability at least $1 - \delta$,*

$$\widehat{\mathcal{E}}(X_{pool}(A_k)) \leq \sum_{i=1}^{k} \frac{n_a^{(i)}}{N_a^{(k)}} \Bigg(\underbrace{\widehat{\mathcal{E}}_a(\hat{h}_i, \hat{t}_i \mid X_v^{(i)})}_{(a)} + \underbrace{\frac{4}{p_0}\sqrt{\frac{2}{n_v^{(i)}}\Big(2d\log\Big(\frac{en_v^{(i)}}{d}\Big) + \log\Big(\frac{8k}{\delta}\Big)\Big)}}_{(b)} + \underbrace{\frac{4}{p_0}\Bigg(\sqrt{\frac{2k}{N_a^{(k)}}\Big(2d\log\Big(\frac{eN_a^{(k)}}{d}\Big) + \log\Big(\frac{8k}{\delta}\Big)\Big)}\Bigg)}_{c}$$

and

$$\widehat{\mathcal{P}}(X_{pool}(A_k)) \geq \sum_{i=1}^{k} \mathbb{P}\big(\mathcal{X}^{(i)}(\hat{h}_i, \hat{t}_i)\big) - 2k\sqrt{\frac{2}{N}\Big(2d\log\Big(\frac{eN}{d}\Big) + \log\Big(\frac{8k}{\delta}\Big)\Big)}.$$

Proof. The proof follows by substituting the Rademacher complexity bounds for finite VC dimension function classes from Lemma A.2.6 in the general result from Theorem 3.3.1.

$$\widehat{\mathcal{E}}(X_{pool}(A_k)) \leq \sum_{i=1}^{k} \frac{n_a^{(i)}}{N_a^{(k)}} \Bigg(\underbrace{\widehat{\mathcal{E}}_a(\hat{h}_i, \hat{t}_i \mid X_v^{(i)})}_{(a)} + \frac{4}{p_0} \underbrace{\big(\mathfrak{R}_{n_v^{(i)}}(\mathcal{H}^{T,g}) + \mathfrak{R}_{n_a^{(i)}}(\mathcal{H}^{T,g})\big)}_{(b)}$$

$$+\frac{4}{p_0}\underbrace{\sqrt{\frac{1}{n_v^{(i)}}\log\left(\frac{8k}{\delta}\right)}}_{(c)}\Bigg) + \underbrace{\frac{4}{p_0}\sqrt{\frac{k}{N_a^{(k)}}\log\left(\frac{8k}{\delta}\right)}}_{(d)}$$

We first simplify the terms dependent on $n_v^{(i)}$ as follows. Here we use the inequality $\sqrt{a}+\sqrt{b} \leq \sqrt{2(a+b)}$ for any $a, b \in \mathbb{R}^+$.

$$\begin{aligned}\mathfrak{R}_{n_v^{(i)}}\left(\mathcal{H}^{T,g}\right) + \sqrt{\frac{1}{n_v^{(i)}}\log\left(\frac{8k}{\delta}\right)} &\leq \sqrt{\frac{2d}{n_v^{(i)}}\log\left(\frac{en_v^{(i)}}{d}\right)} + \sqrt{\frac{1}{n_v^{(i)}}\log\left(\frac{4k}{\delta}\right)}, \\ &\leq \sqrt{\frac{2}{n_v^{(i)}}\left(2d\log\left(\frac{en_v^{(i)}}{d}\right) + \log\left(\frac{8k}{\delta}\right)\right)}.\end{aligned}$$

Next, we simplify the terms dependent on $n_a^{(i)}$ as follows. First, we substitute the Rademacher complexity using the bound in Lemma A.2.6 and then apply the same steps as in the proof of Theorem 3.3.1 to bound $\sum_{i=1}^{k}\sqrt{n_a^{(i)}}/N_a^{(k)}$ by $\sqrt{k/N_a^{(k)}}$ followed by the application of $\sqrt{a}+\sqrt{b} \leq \sqrt{2(a+b)}$ to get the final term.

$$\begin{aligned}\sum_{i=1}^{k}\frac{n_a^{(i)}}{N_a^{(k)}}\mathfrak{R}_{n_a^{(i)}}\left(\mathcal{H}^{T,g}\right) + \sqrt{\frac{k}{N_a^{(k)}}\log(\frac{8k}{\delta})} &\leq \sum_{i=1}^{k}\frac{n_a^{(i)}}{N_a^{(k)}}\sqrt{\frac{2d}{n_a^{(i)}}\log\left(\frac{en_a^{(i)}}{d}\right)} + \sqrt{\frac{k}{N_a^{(k)}}\log\left(\frac{8k}{\delta}\right)} \\ &= \sum_{i=1}^{k}\frac{\sqrt{n_a^{(i)}}}{N_a^{(k)}}\sqrt{2d\log\left(\frac{en_a^{(i)}}{d}\right)} + \sqrt{\frac{k}{N_a^{(k)}}\log\left(\frac{8k}{\delta}\right)} \\ &\leq \sum_{i=1}^{k}\frac{\sqrt{n_a^{(i)}}}{N_a^{(k)}}\sqrt{2d\log\left(\frac{eN_a^{(k)}}{d}\right)} + \sqrt{\frac{k}{N_a^{(k)}}\log\left(\frac{8k}{\delta}\right)} \\ &\leq \sqrt{\frac{2dk}{N_a^{(k)}}\log\left(\frac{eN_a^{(k)}}{d}\right)} + \sqrt{\frac{k}{N_a^{(k)}}\log\left(\frac{8k}{\delta}\right)} \\ &\leq \sqrt{\frac{2k}{N_a^{(k)}}\left(2d\log\left(\frac{eN_a^{(k)}}{d}\right) + \log\left(\frac{8k}{\delta}\right)\right)}.\end{aligned}$$

□

A.2.3 Homogeneous Linear Classifiers with Uniform Distribution

Here we instantiate Theorem 3.3.1 for the case of homogeneous linear separators under the uniform distribution in the realizable setting. Formally, let $P_{\mathbf{x}}$ be a uniform distribution supported on the unit ball in $\mathbb{R}^d$, $\mathcal{X} = \{\mathbf{x} \in \mathbb{R}^d : ||\mathbf{x}|| \leq 1\}$. Let $\mathcal{W} = \{\mathbf{w} \in \mathbb{R}^d : ||\mathbf{w}||_2 = 1\} = \mathbb{S}_d$ and $\mathcal{H} = \{\mathbf{x} \mapsto \text{sign}(\langle \mathbf{w}, \mathbf{x} \rangle) \, \forall \mathbf{w} \in \mathcal{W}\}$, the score function is given by $g(h, \mathbf{x}) = g(\mathbf{w}, \mathbf{x}) = |\langle \mathbf{w}, \mathbf{x} \rangle|$ and set $T = [0, 1]$. For simplicity, we will use $\mathcal{W}$ in place of $\mathcal{H}$.

Corollary 3.3.2.1. *(Overall Auto-Labeling Error and Coverage) Let $\hat{\mathbf{w}}_i, \hat{t}_i$ be the ERM solution and the auto-labeling margin threshold respectively at epoch i. Let $n_v^{(i)}, n_a^{(i)}$ denote the number of validation and auto-labeled points at epoch i. Let the auto-labeling algorithm run for k-epochs. Then, for any $\delta \in (0, 1)$, w.p. at least $1 - \delta/2$,*

$$\widehat{\mathcal{E}}(X_{pool}(A_k)) \leq \sum_{i=1}^{k} \frac{n_a^{(i)}}{N_a^{(k)}} \Bigg(\underbrace{\widehat{\mathcal{E}}_a(\hat{\mathbf{w}}_i, \hat{t}_i \mid X_v^{(i)})}_{(a)} + \underbrace{\frac{4}{p_0} \sqrt{\frac{2}{n_v^{(i)}} \left(2d \log\left(\frac{e n_v^{(i)}}{d} \right) + \log\left(\frac{8k}{\delta} \right) \right)}}_{(b)} + \underbrace{\frac{4}{p_0} \left(\sqrt{\frac{2k}{N_a^{(k)}} \left(2d \log\left(\frac{e N_a^{(k)}}{d} \right) + \log\left(\frac{8k}{\delta} \right) \right)} \right)}_{c} \Bigg)$$

and w.p. at least $1 - \delta/2$

$$\widehat{\mathcal{P}}(X_{pool}(A_k)) \geq 1 - \min_i \hat{t}_i \sqrt{4d/\pi} - 2k \sqrt{\frac{2}{N} \left(2d \log\left(\frac{eN}{d} \right) + \log\left(\frac{8k}{\delta} \right) \right)}.$$

Proof. The bound on auto-labeling error follows directly from Theorem A.2.6.1 as the VC dimension for this setting is d. For the coverage bound, we utilize the fact that the distribution $P_{\mathbf{x}}$ is the uniform distribution over the unit ball. This enables us to obtain explicit lower bounds on the coverage. The details are given in Lemma A.2.7 and Lemma A.2.8. □

Lemma A.2.7. *Let the auto-labeling algorithm run for k-epochs and let $\hat{\mathbf{w}}_i, \hat{t}_i$ be the ERM solution and the auto-labeling margin threshold respectively at epoch i. Let $\mathcal{X}^{(i)}$ be the unlabeled*

region at the beginning of epoch i, then we have,

$$\sum_{i=1}^{k} \mathbb{P}\big(\mathcal{X}^{(i)}(\hat{\mathbf{w}}_i, \hat{t}_i)\big) \geq 1 - \min_i \hat{t}_i \sqrt{4d/\pi}. \tag{A.13}$$

Proof. Let $\mathcal{X}(\hat{\mathbf{w}}_i, t_i) = \{\mathbf{x} \in \mathcal{X} : |\langle \hat{\mathbf{w}}_i, \mathbf{x} \rangle| \geq \hat{t}_i\}$ denote the region that can be auto-labeled by $\hat{\mathbf{w}}_i, \hat{t}_i$. However, since in each round the remaining region is $\mathcal{X}^{(i)}$ the actual auto-labeled region of epoch i is $\mathcal{X}_a^{(i)} = \{\mathbf{x} \in \mathcal{X}^{(i)} : |\langle \hat{\mathbf{w}}_i, \mathbf{x} \rangle| \geq \hat{t}_i\}$. Let $\bar{\mathcal{X}}(\hat{\mathbf{w}}_i, t_i)$ denote the complement of set $\mathcal{X}(\hat{\mathbf{w}}_i, t_i)$.

Now observe that $\mathcal{X}_a = \cup_{i=1}^{k} \mathcal{X}_a^{(i)}$ and $\mathcal{X}(\hat{\mathbf{w}}_k, \hat{t}_k) \subseteq \mathcal{X}_a$ because any $\mathbf{x} \in \mathcal{X}(\hat{\mathbf{w}}_k, \hat{t}_k)$ is either auto-labeled in previous rounds $i < k$ or if not then it will be auto-labeled in the k^{th} round. More specifically, any $\mathbf{x} \in \mathcal{X}(\hat{\mathbf{w}}_k, \hat{t}_k)$ is either in $\cup_{i=1}^{k-1} \mathcal{X}_a^{(i)}$ and if not then it must be in $\mathcal{X}_a^{(k)}$. Thus the sum of probabilities,

$$\begin{aligned}
\sum_{i=1}^{k} \mathbb{P}\big(\mathcal{X}^{(i)}(\hat{\mathbf{w}}_i, \hat{t}_i)\big) &= \sum_{i=1}^{k} \mathbb{P}(\mathcal{X}_a^{(i)}) \\
&= \mathbb{P}(\mathcal{X}_a) \\
&\geq \min_i \mathbb{P}\big(\mathcal{X}(\hat{\mathbf{w}}_i, \hat{t}_i)\big) \\
&= 1 - \max_i \mathbb{P}\big(\bar{\mathcal{X}}(\hat{\mathbf{w}}_i, \hat{t}_i)\big) \\
&\geq 1 - \min_i \hat{t}_i \sqrt{4d/\pi}
\end{aligned}$$

The last step used Lemma 4 from (Balcan et al., 2007)) with $\gamma_1 = \hat{t}_i$ and $\gamma_2 = 0$ to upper bound $\mathbb{P}(\bar{\mathcal{X}}(\hat{\mathbf{w}}_i, \hat{t}_i))$ by $\hat{t}_i \sqrt{4d/\pi}$. The lemma is stated as follows in Lemma A.2.8, □

Lemma A.2.8. *((Balcan et al., 2007) (Lemma 4)) Let $d \geq 2$ and let $\mathbf{x} = [x_1, \ldots x_d]$ be uniformly distributed in the d-dimensional unit ball. Given $\gamma_1 \in [0, 1], \gamma_2 \in [0, 1]$, we have:*

$$\mathbb{P}\big((x_1, x_2) \in [0, \gamma_1] \times [\gamma_2, 1]\big) \leq \frac{\gamma_1 \sqrt{d}}{2\sqrt{\pi}} \exp\Big(-\frac{(d-2)\gamma_2^2}{2}\Big)$$

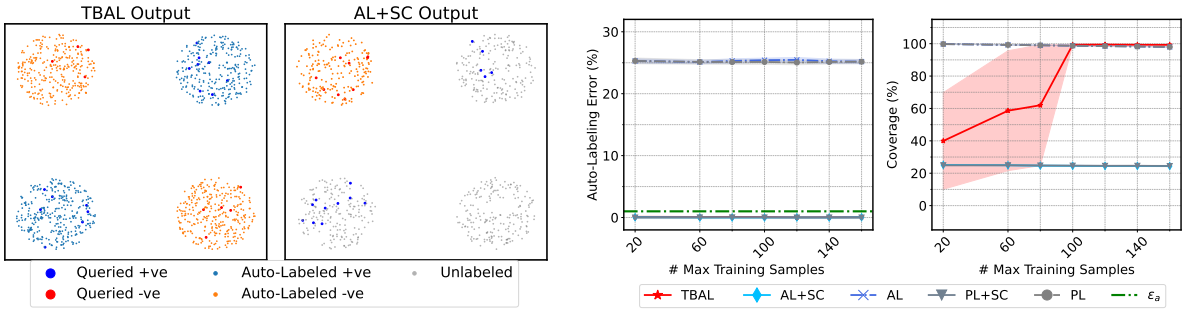
A.2.4 Lower Bound

Lemma 3.3.2. *Let c_1, c_2 and $\sigma > 0$. Let $\mathbf{x}_i \in X$ be a set of n i.i.d. points from $\mathcal{X}$ with corresponding true labels y_i. Given $(h,t) \in \mathcal{H}^{T,g}$, let $\mathbb{E}\big[\big(\ell_{0-1}^{\perp}(h,t,\mathbf{x}_i,y_i) - \mathcal{E}(h,t \mid \mathcal{X})\big)^2\big] = \sigma_i^2 > \sigma^2$ for every $\mathbf{x}_i$ for $\sigma_i > 0$ and let $\sum_i^n \sigma_i^2 \geq c_1$ then for every $\epsilon \in [0, \frac{\sum_{i=1}^n \sigma_i^2}{\sqrt{c_1}}]$ with $n_v < \frac{12\sigma^2}{\epsilon^2}\log(4c_2)$ the following holds w.p. at least $1/4$, $\mathcal{E}_a(h,t \mid \mathcal{X}) > \widehat{\mathcal{E}}_a(h,t \mid X) + \epsilon$.*

Proof. It follows by application of Feller's result stated in lemma A.2.9. □

Lemma A.2.9. *(Feller, Lower Bound on Tail Probability of Sum of Independent Random Variables) There exists positive universal constants c_1 and c_2 such that for any set of independent random variables $X_1, \ldots, X_m$ satisfying $E[X_i] = 0$ and $|X_i| \leq M$, for every $i \in \{1, \ldots, m\}$, if $\sum_{i=1}^m \mathbb{E}[(X)_i^2] \geq c_1$, then for every $\epsilon \in [0, \frac{\sum_{i=1}^m \mathbb{E}[(X_i)^2]\}}{M\sqrt{c_1}}]$*

$$\mathbb{P}(\sum_{i=1}^{m} X_i > \epsilon) \geq c_2 \exp\Big(\frac{-\epsilon^2}{12\sum_{i=1}^m \mathbb{E}[(X_i)^2]}\Big). \tag{A.14}$$



(a) Output of TBAL and AL+SC on XOR dataset

(b) Auto labeling performance of various methods

Figure A.1: Comparison of Threshold-Based Auto-Labeling (TBAL) and Active-Learning followed by Selective Classification (AL+SC) on XOR-dataset. Left figure (a) shows samples that were auto-labeled, queried, and left unlabeled by these methods. Figure (b) shows the auto-labeling error and coverage achieved. The lines show the mean, and the shaded region shows 1-standard deviation estimated over 10 trials with different random seeds.

A.3 Additional Experiments

In this section, we discuss additional experiments on the role of hypothesis class in auto-labeling datasets and experiments for studying the role of confidence function in auto-labeling. Finally, we visualize PaCMAP embeddings of the CIFAR-10 and MNIST data points to get a sense of auto-labeling regions in various rounds of the algorithm.

A.3.1 Additional Experiments on Role of the Hypothesis Class

First, we provide details of the datasets,

XOR is a synthetic dataset. Recall that it is created by uniformly drawing points from 4 circles, each centered at the corners of a square of with side length 4 centered at the origin. Points in the diagonally opposite balls belong to the same class. We generate a total of $N = 10,000$ samples, out of which we keep $8,000$ in X_{pool} and $2,000$ in the validation pool X_{val}.

MNIST (Deng, 2012) is a standard image dataset of hand-written digits. We randomly split the standard training set into X_{pool} and the validation pool X_{val} of sizes 48,000 and 12,000, respectively. While training a linear classifier on this dataset, we flatten the 28×28 images to vectors of size 784.

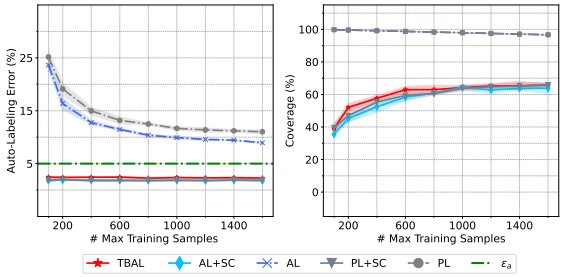
XOR Experiment. We run the TBAL algorithm 1 with an error tolerance of $\epsilon_a = 1\%$. we use 20% of N_q as seed training data and keep query size n_b as 5% of N_q. We compare it with active learning and active learning followed by selective classification. The given function class and selective classifier are both linear for all the algorithms. The results are shown in Figure A.1. Clearly, there is no linear classifier that can correctly classify this data. We note that there are multiple optimal classifiers in the function class of linear classifiers, and they will all incur an error of 25%. So, active learning algorithms can only output models that make at least 25% error. If we naively use the output model for auto-labeling, we can obtain near full coverage but incur 25% auto-labeling error. If we use the model output by active learning with threshold-based selective classification, then it can attain a lower error in labeling. However, it can only label $\approx 25\%$ of the unlabeled data. In contrast, the TBAL algorithm can label almost all of the data accurately, i.e., attain close to 100% coverage, with an error close to 1% auto-labeling error.

MNIST Experiment. For training LeNet (LeCun et al., 1998), we use SGD with a learning rate of 0.1, batch size of 32, and train for 20 epochs. We use the auto-labeling error threshold $\epsilon_a = 5\%$. We use 20% of N_q as seed training data and keep query size n_b as 5% of N_q. The results are presented in Figure A.2 we observe that TBAL using less powerful models can still yield highly accurate datasets with a significant fraction of points labeled by the models. This confirms the notion that bad models can still provide good datasets.

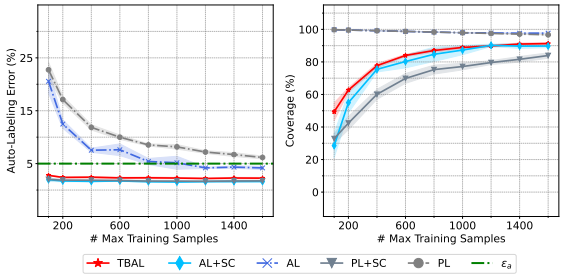
A.3.2 Role of Confidence Function

The confidence function g is used to obtain uncertainty scores is an important factor in auto-labeling. In particular, for threshold-based auto-labeling, we expect the scores of correctly classified and incorrectly classified points to be reasonably well separated, and if this is not the case, then the algorithm will struggle to find a good threshold, even if the given classifier has good accuracy in certain regions.

Setup. We perform auto-labeling on the CIFAR-10 dataset using a small CNN network with 2 convolution layers followed by 3 fully connected layers (PyTorch, 2022). We use two different scores for auto-labeling: a) Usual softmax output, b) Energy score with temperature = 1 (LeCun



(a) Auto-labeling MNIST data using a linear classifier. The validation size used here is 12k.

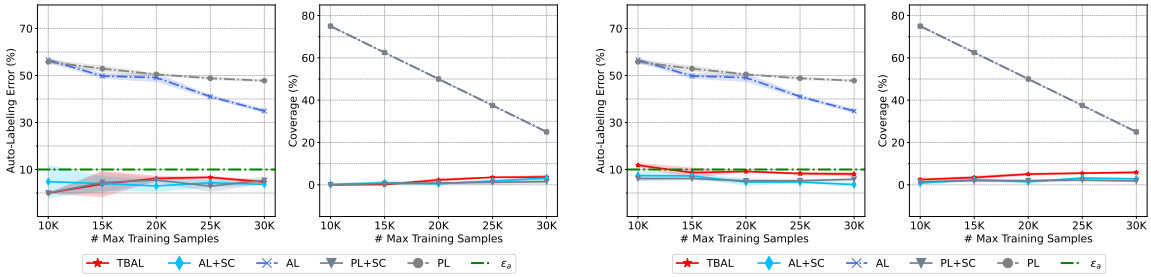


(b) Auto-labeling MNIST data using LeNet classifier. The validation size used here is 12k.

Figure A.2: Auto-labeling performance on MNIST data using different models (hypothesis classes) as a function of samples available for training. The left figure (a) shows the results with the linear classifier, and the right figure (b) shows the results with the LeNet classifier. The auto-labeling error threshold $\epsilon_a = 5\%$ in both experiments, and the algorithms are given the same amount of validation data. The lines show the mean, and the shaded region shows 1-standard deviation estimated over 5 trials with different random seeds.

et al., 2006). We vary the maximum number of training samples N_q and keep 20% of N_q as seed samples and query points in batches of 10% of N_q. The model is trained for 50 epochs, using SGD with a learning rate of 0.05, batch size = 256, weight decay = $5e^{-4}$, and momentum=0.9, and uses $\epsilon_a = 10\%$.

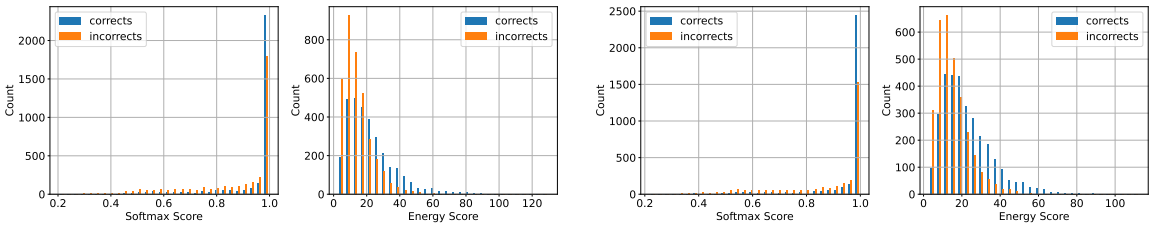
Results. The results with softmax scores and energy scores used as confidence functions can be seen in Figures A.3(a) and A.3(b), respectively. We see that for both of these cases, TBAL does not obtain a coverage of more than $\approx 6\%$. We observe that using the energy score as the confidence function performed marginally better than using the softmax scores. We note that this is the case even though the test accuracies of the trained models were around 50% for most of the rounds. Note that CIFAR-10 has 10 classes, so an accuracy of 50% is much better than random guessing, and one would expect to be able to auto-label a significant chunk of the data with such a model. However, the softmax scores and energy scores are not well calibrated, and therefore, when used as confidence functions, they result in a poor separation between correct and incorrect predictions by the model. This can be seen in Figure A.4 where neither of the softmax and energy scores provides a good separation between the correct and incorrect predictions. We can also see that the energy score is marginally better in terms of the separation, which allows it to achieve slightly better auto-labeling coverage in comparison to using softmax scores. This suggests



(a) Auto-labeling CIFAR-10 data using a small network and softmax scores. Validation size = 10k.

(b) Auto-labeling CIFAR-10 data using a small network and energy scores. Validation size = 10k.

Figure A.3: Auto-labeling performance on CIFAR-10 data using a small network and different scoring functions. The left figure (a) shows the results with softmax scores, and the right figure (b) shows the results with the energy score. The auto-labeling error threshold $\epsilon_a = 10\%$ in both experiments and methods are given the same amount of validation data. The lines show the mean, and the shaded region shows 1-standard deviation estimated over 5 trials with different random seeds.



(a) Histogram of scores in round 2.

(b) Histogram of scores in round 6.

Figure A.4: Histograms of scores computed on the validation data in a few rounds of TBAL run on CIFAR-10 with a small net. We picked two rounds where it auto-labeled the most, i.e., around 800 points.

that more investigation is needed to understand the properties of good confidence functions for auto-labeling, which is left to future work. For a more detailed visualization of the rounds of TBAL for this experiment, see Figures A.6 and A.7 in the Appendix.

A.3.3 Detailed Results

In the main paper, we omitted AL, PL, and PL+SC for clarity due to a lack of space. We provide results including these baselines in the following tables for IMDB, Tiny-ImageNet, and Unit-Ball datasets.

A.3.4 Auto Labeling Visualization

In this section, we visualize the process of TBAL. We use the dimensionality reduction method, PaCMAP Wang et al. (2021c), to visualize the features of the samples. For neural network models, we visualize the PaCMAP embeddings of the penultimate layer's output and for linear models, we use PaCMAP on the raw features. In these figures, each row corresponds to one TBAL round. Each figure shows a few selected rounds of auto-labeling. Each figure has four columns (left to right), which show: **a**) The samples that are labeled by TBAL in the round are shown in that row. **b**) The embeddings for training samples in that round. **c**) The embeddings for validation data points in that round. **d**) The score distribution for the validation dataset in that round.

In Figure A.5 we see visualizations for auto-labeling on the MNIST data using linear models. In this setting the data exhibits clustering structure in the PaCMAP embeddings learned on the raw features and the confidence (probability) scores produced are also reasonably well calibrated which leads to good auto-labeling performance.

The visualizations for the process of TBAL on CIFAR-10 using the small network (a small CNN network with 2 convolution layers followed by 3 fully connected layers PyTorch (2022)) with energy scores and soft-max scores for confidence functions are shown in Figures A.6 and A.7 respectively. We note that both the energy scores and soft-max scores do not seem to be calibrated to the correctness of the predicted labels, which makes it difficult to identify subsets of unlabeled data where the current hypothesis in each round could have potentially auto-labeled. We also note that the test accuracies of the trained models were around 50% for most of the rounds of TBAL, even though the small network model is not a powerful enough model class for this dataset. Note that CIFAR-10 has 10 classes, so the accuracy of 50% is much better than random guessing, and one would expect to be able to auto-label a reasonably large chunk of the data with such a model if accompanied by a good confidence function. This highlights the important role that the confidence function plays in a TBAL system, and further investigation is needed, which is left to future work.

Note that, in our auto-labeling implementation, we find class specific thresholds. In these figures,

we show the histograms of scores for all classes for simplicity. We want to emphasize that the visualization figures in this section are 2D representations (approximation) of the high-dimensional features (either of the penultimate layer or the raw features).

Symbol	Definition
$\mathcal{X}$	feature space.
$\mathcal{Y}$	label space.
$\mathcal{H}$	hypothesis space.
h	a hypothesis in $\mathcal{H}$.
$\mathbf{x}, y$	$\mathbf{x}$ is an element in $\mathcal{X}$ and y is its true label.
$\mathcal{S}, \mathrm{S}'$	$\mathcal{S} \subseteq \mathcal{X}$ is a sub-region in $\mathcal{X}$, $\mathrm{S}' = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$ i.i.d. samples in $\mathcal{S}$.
X_{pool}	unlabeled pool of data points.
$X_v^{(i)}, n_v^{(i)}$	set of validation points at the beginning of ith round and $n_v^{(i)} = \lvert X_v^{(i)} \rvert$.
$X_a^{(i)}, n_a^{(i)}$	set of auto-labeled points in ith round and $n_a^{(i)} = \lvert X_a^{(i)} \rvert$.
$\hat{h}_i, \hat{t}_i$	ERM solution and auto-labeling thresholds respectively in ith round.
$\mathcal{X}^{(i)}$	unlabeled region left at the beginning of ith round.
$X^{(i)}$	unlabeled pool left at the beginning of ith round.
$m_a^{(i)}$	number of auto-labeling mistakes in ith round.
k	number of rounds of the TBAL algorithm.
$X_{pool}(A_k)$	set of all auto-labeled points till the end of round k.
g	confidence function $g : \mathcal{H} \times \mathcal{X} \mapsto T$. Where $T \subseteq \mathbb{R}^+$, usually $T = [0, 1]$
$\mathcal{H}^{T,g}$	Cartesian product of $\mathcal{H}$ and T the range of g.
$N_a^{(k)}$	$\sum_{i=1}^{k} n_a^{(i)}$.
$\ell_{0-1}(h, \mathbf{x}, y)$	$\mathbb{1}(h(\mathbf{x}) \neq y)$.
$\ell_{\perp}(h, t, \mathbf{x})$	$\mathbb{1}(g(h, \mathbf{x}) \geq t)$.
$\ell_{0-1}^{\perp}(h, t, \mathbf{x}, y)$	$\ell_{0-1}(h, \mathbf{x}, y) \cdot \ell_{\perp}((h, t), \mathbf{x})$.
$\mathfrak{R}_n(\mathcal{H}, \ell_{0-1})$	$\mathbb{E}_{\sigma,S}\left[\sup_{h\in\mathcal{H}} \frac{1}{n}\sum_{i=1}^{n} \sigma_i \ell_{0-1}(h, \mathbf{x}_i, y_i)\right]$.
$\mathfrak{R}_n(\mathcal{H}^{T,g}, \ell_{\perp})$	$\mathbb{E}_{\sigma,S}\left[\sup_{(h,t)\in\mathcal{H}^{T,g}} \frac{1}{n}\sum_{i=1}^{n} \sigma_i \ell_{\perp}(h, t, \mathbf{x}_i)\right]$.
$\mathfrak{R}_n(\mathcal{H}^{T,g}, \ell_{0-1}^{\perp})$	$\mathbb{E}_{\sigma,S}\left[\sup_{(h,t)\in\mathcal{H}^{T,g}} \frac{1}{n}\sum_{i=1}^{n} \sigma_i \ell_{0-1}^{\perp}(h, t, \mathbf{x}_i, y_i)\right]$.
$\mathfrak{R}_n(\mathcal{H}^{T,g})$	$\mathfrak{R}_n(\mathcal{H}, \ell_{0-1}) + \mathfrak{R}_n(\mathcal{H}^{T,g}, \ell_{\perp})$.
$\mathcal{E}(h, t \mid \mathcal{S})$	$\mathbb{E}_{\mathbf{x}\mid\mathcal{S}}[\ell_{0-1}^{\perp}(h, t, \mathbf{x}, y)]$.
$\widehat{\mathcal{E}}(h, t \mid \mathrm{S}')$	$\frac{1}{\lvert\mathrm{S}'\rvert}\sum_{i=1}^{\lvert\mathrm{S}'\rvert} \ell_{0-1}^{\perp}(h, t, \mathbf{x}_i, y_i)$.
$\mathbb{P}(h, t \mid \mathcal{S})$	$\mathbb{E}_{\mathbf{x}\mid\mathcal{S}}[\ell_{\perp}(h, t, \mathbf{x})]$.
$\widehat{\mathbb{P}}(h, t \mid \mathrm{S}')$	$\frac{1}{\lvert\mathrm{S}'\rvert}\sum_{i=1}^{\lvert\mathrm{S}'\rvert} \ell_{\perp}(h, t, \mathbf{x}_i, y_i)$.
$\mathcal{E}_a(h, t \mid \mathcal{S})$	$\mathcal{E}(h, t \mid \mathcal{S})/\mathbb{P}(h, t \mid \mathcal{S})$.
$\widehat{\mathcal{E}}_a(h, t \mid \mathrm{S}')$	$\widehat{\mathcal{E}}(h, t \mid \mathrm{S}')/\widehat{\mathbb{P}}(h, t \mid \mathrm{S}')$.

Table A.1: Glossary of variables and symbols used in this paper.

N_v	Error (%)					Coverage (%)				
	TBAL	**AL+SC**	**PL+SC**	**AL**	**PL**	**TBAL**	**AL+SC**	**PL+SC**	**AL**	**PL**
200	4.77 ±0.18	3.35 ±0.80	3.68 ±0.38	6.60 ±1.06	6.16 ±0.10	83.14 ±3.65	78.53 ±7.05	81.20 ±1.78	98.75 ±0.00	97.50 ±0.00
400	4.57 ±0.26	3.53 ±0.73	3.96 ±0.33	6.60 ±1.06	6.16 ±0.10	90.70 ±3.11	86.39 ±5.11	87.43 ±2.01	98.75 ±0.00	97.50 ±0.00
600	4.32 ±0.17	3.70 ±0.63	4.17 ±0.27	6.60 ±1.06	6.16 ±0.10	92.96 ±0.46	88.90 ±4.83	90.12 ±1.62	98.75 ±0.00	97.50 ±0.00
800	4.66 ±0.20	3.84 ±0.70	4.15 ±0.35	6.60 ±1.06	6.16 ±0.10	92.42 ±0.89	88.67 ±3.88	89.37 ±1.45	98.75 ±0.00	97.50 ±0.00
1000	4.67 ±0.16	3.90 ±0.68	4.21 ±0.35	6.60 ±1.06	6.16 ±0.10	92.89 ±0.91	89.79 ±3.09	90.18 ±1.39	98.75 ±0.00	97.50 ±0.00

Table A.2: **IMDB**. Effect of variation of validation data size (N_v) without using a UCB (i.e., $C_1 = 0$) on error estimates. We keep training data size N_q fixed at 500 and use error threshold $\epsilon_a = 5\%$. We report the mean and std. deviation over 10 runs with different random seeds.

N_v	Error (%)					Coverage (%)				
	TBAL	**AL+SC**	**PL+SC**	**AL**	**PL**	**TBAL**	**AL+SC**	**PL+SC**	**AL**	**PL**
200	2.28 ±0.21	3.11 ±0.86	2.86 ±0.35	6.60 ±1.06	6.16 ±0.10	68.24 ±6.20	57.77 ±13.09	60.45 ±1.63	98.75 ±0.00	97.50 ±0.00
400	1.29 ±0.10	1.98 ±0.40	1.54 ±0.11	6.60 ±1.06	6.16 ±0.10	63.81 ±4.86	63.06 ±10.70	68.32 ±6.60	98.75 ±0.00	97.50 ±0.00
600	1.41 ±0.20	1.81 ±0.22	1.87 ±0.07	6.60 ±1.06	6.16 ±0.10	69.64 ±3.98	62.92 ±9.20	69.84 ±3.07	98.75 ±0.00	97.50 ±0.00
800	1.62 ±0.30	2.04 ±0.35	2.33 ±0.35	6.60 ±1.06	6.16 ±0.10	67.45 ±3.72	63.22 ±7.89	72.50 ±2.28	98.75 ±0.00	97.50 ±0.00
1000	1.64 ±0.23	1.97 ±0.26	1.93 ±0.13	6.60 ±1.06	6.16 ±0.10	70.28 ±2.82	66.11 ±8.00	73.04 ±2.06	98.75 ±0.00	97.50 ±0.00

Table A.3: **IMDB**. Effect of variation of validation data size (N_v), using a UCB (i.e., $C_1 = 0.25$) on error estimates. We keep training data size N_q fixed at 500 and use error threshold $\epsilon_a = 5\%$. We report the mean and std. deviation over 10 runs with different random seeds.

N_q	Error (%)					Coverage (%)				
	TBAL	**AL+SC**	**PL+SC**	**AL**	**PL**	**TBAL**	**AL+SC**	**PL+SC**	**AL**	**PL**
200	4.57 ±0.21	4.30 ±0.20	3.97 ±0.06	6.44 ±0.20	6.20 ±0.09	93.46 ±1.01	90.60 ±2.91	91.97 ±0.28	99.50 ±0.00	99.00 ±0.00
400	4.60 ±0.09	3.75 ±0.83	3.92 ±0.86	11.86 ±13.54	6.81 ±1.26	92.55 ±0.66	84.27 ±19.15	89.95 ±2.92	99.00 ±0.00	98.00 ±0.00
600	4.93 ±0.10	3.99 ±0.91	4.69 ±0.09	6.31 ±1.28	6.33 ±0.10	92.45 ±0.84	91.69 ±3.99	91.20 ±0.42	98.50 ±0.00	97.00 ±0.00
800	4.76 ±0.12	3.55 ±0.69	4.37 ±0.14	6.91 ±1.49	6.12 ±0.10	92.15 ±1.05	89.98 ±3.38	89.97 ±0.69	98.00 ±0.00	96.00 ±0.00
1000	4.49 ±0.06	4.19 ±0.31	4.25 ±0.29	5.65 ±0.25	6.14 ±0.11	92.25 ±0.96	92.28 ±2.13	89.47 ±0.70	97.50 ±0.00	95.00 ±0.00

Table A.4: **IMDB**. Effect of variation of N_q, the maximum number of samples the algorithm can use for training, without using a UCB (i.e., $C_1 = 0$) on error estimates. We keep validation data size N_v fixed at 1000 and use error threshold $\epsilon_a = 5\%$. We report the mean and std. deviation over 10 runs with different random seeds.

N_q	Error (%)					Coverage (%)				
	TBAL	AL+SC	PL+SC	AL	PL	TBAL	AL+SC	PL+SC	AL	PL
200	1.67 ±0.29	2.15 ±0.45	1.59 ±0.10	6.44 ±0.20	6.20 ±0.09	73.30 ±3.49	57.17 ±11.09	57.39 ±4.15	99.50 ±0.00	99.00 ±0.00
400	1.63 ±0.19	1.61 ±0.29	1.76 ±0.13	11.86 ±13.54	6.81 ±1.26	72.59 ±3.16	64.53 ±16.61	58.48 ±1.79	99.00 ±0.00	98.00 ±0.00
600	1.67 ±0.21	1.83 ±0.30	1.67 ±0.08	6.31 ±1.28	6.33 ±0.10	71.38 ±2.15	70.50 ±5.68	65.71 ±2.14	98.50 ±0.00	97.00 ±0.00
800	1.67 ±0.27	1.90 ±0.31	1.79 ±0.09	6.91 ±1.49	6.12 ±0.10	69.10 ±4.51	65.74 ±10.14	73.21 ±2.57	98.00 ±0.00	96.00 ±0.00
1000	1.62 ±0.22	1.97 ±0.35	1.70 ±0.12	5.65 ±0.25	6.14 ±0.11	73.42 ±2.84	68.05 ±5.56	64.18 ±2.11	97.50 ±0.00	95.00 ±0.00

Table A.5: **IMDB**. Effect of variation of N_q, the maximum number of samples the algorithm can use for training, using a UCB (i.e., $C_1 = 0.25$) on error estimates. We keep validation data size N_v fixed at 1000 and use error threshold $\epsilon_a = 5\%$. We report the mean and std. deviation over 10 runs with different random seeds.

N_v	Error (%)					Coverage (%)				
	TBAL	AL+SC	PL+SC	AL	PL	TBAL	AL+SC	PL+SC	AL	PL
100	3.10 ±1.80	0.68 ±0.81	1.45 ±0.73	1.23 ±0.99	2.87 ±0.57	71.43 ±8.86	96.95 ±1.01	92.29 ±3.27	98.52 ±0.16	96.88 ±0.00
400	1.97 ±0.76	0.59 ±0.18	1.19 ±0.53	0.81 ±0.26	2.87 ±0.57	93.99 ±2.39	97.89 ±0.50	91.73 ±2.86	98.44 ±0.00	96.88 ±0.00
800	1.64 ±0.50	0.66 ±0.19	1.21 ±0.41	0.81 ±0.26	2.87 ±0.57	96.26 ±1.33	98.06 ±0.53	92.25 ±2.31	98.44 ±0.00	96.88 ±0.00
1200	1.39 ±0.39	0.67 ±0.19	1.11 ±0.30	0.81 ±0.26	2.87 ±0.57	96.67 ±0.84	98.10 ±0.45	91.98 ±2.20	98.44 ±0.00	96.88 ±0.00
1600	1.33 ±0.30	0.70 ±0.19	1.11 ±0.26	0.81 ±0.26	2.87 ±0.57	97.13 ±0.45	98.16 ±0.44	92.01 ±2.09	98.44 ±0.00	96.88 ±0.00
2000	1.28 ±0.34	0.71 ±0.21	1.07 ±0.25	0.81 ±0.26	2.87 ±0.57	97.15 ±0.54	98.20 ±0.34	91.86 ±2.17	98.44 ±0.00	96.88 ±0.00

Table A.6: **Unit Ball**. Effect of variation of validation data size (N_v), without using a UCB (i.e., $C_1 = 0$) on error estimates. We keep training data size N_q fixed at 500 and use error threshold $\epsilon_a = 1\%$. We report the mean and std. deviation over 10 runs with different random seeds.

N_v	Error (%)					Coverage (%)				
	TBAL	AL+SC	PL+SC	AL	PL	TBAL	AL+SC	PL+SC	AL	PL
100	3.10 ±1.80	0.68 ±0.81	1.45 ±0.73	1.23 ±0.99	2.87 ±0.57	71.43 ±8.86	96.95 ±1.01	92.29 ±3.27	98.52 ±0.16	96.88 ±0.00
400	1.65 ±0.65	0.32 ±0.15	0.52 ±0.32	0.81 ±0.26	2.87 ±0.57	93.27 ±2.50	96.91 ±0.99	87.86 ±3.73	98.44 ±0.00	96.88 ±0.00
800	1.08 ±0.47	0.24 ±0.16	0.31 ±0.17	0.81 ±0.26	2.87 ±0.57	96.01 ±1.16	96.31 ±1.36	86.21 ±3.55	98.44 ±0.00	96.88 ±0.00
1200	0.78 ±0.27	0.17 ±0.11	0.18 ±0.14	0.81 ±0.26	2.87 ±0.57	96.82 ±0.84	95.96 ±1.40	84.65 ±4.14	98.44 ±0.00	96.88 ±0.00
1600	0.65 ±0.20	0.13 ±0.08	0.12 ±0.09	0.81 ±0.26	2.87 ±0.57	96.93 ±0.57	95.70 ±1.38	83.76 ±3.93	98.44 ±0.00	96.88 ±0.00
2000	0.54 ±0.16	0.21 ±0.11	0.21 ±0.10	0.81 ±0.26	2.87 ±0.57	97.23 ±0.42	96.36 ±1.13	85.72 ±3.47	98.44 ±0.00	96.88 ±0.00

Table A.7: **Unit-Ball**. Effect of variation of validation data size (N_v), without using a UCB (i.e., $C_1 = 0.25$) on error estimates. We keep training data size N_q fixed at 500 and use error threshold $\epsilon_a = 1\%$. We report the mean and std. deviation over 10 runs with different random seeds.

N_q	Error (%)					Coverage (%)				
	TBAL	**AL+SC**	**PL+SC**	**AL**	**PL**	**TBAL**	**AL+SC**	**PL+SC**	**AL**	**PL**
100	1.53 ±0.27	1.16 ±0.35	1.14 ±0.29	16.93 ±2.48	12.53 ±2.24	75.31 ±7.06	31.69 ±10.78	51.47 ±10.46	99.69 ±0.00	99.38 ±0.00
200	1.25 ±0.21	1.04 ±0.25	0.98 ±0.17	7.85 ±1.69	7.08 ±1.82	96.24 ±0.88	73.27 ±8.68	76.05 ±8.19	99.38 ±0.00	98.75 ±0.00
400	1.20 ±0.20	0.94 ±0.17	1.03 ±0.15	1.81 ±0.58	3.25 ±0.81	97.70 ±0.19	96.48 ±1.74	91.08 ±2.63	98.75 ±0.00	97.50 ±0.00
600	1.21 ±0.26	0.44 ±0.13	1.09 ±0.18	0.44 ±0.14	2.25 ±0.56	97.56 ±0.20	98.11 ±0.02	93.19 ±1.60	98.12 ±0.00	96.25 ±0.00
800	1.13 ±0.20	0.12 ±0.05	1.02 ±0.18	0.12 ±0.05	1.76 ±0.50	97.25 ±0.19	97.49 ±0.01	93.23 ±1.01	97.50 ±0.00	95.00 ±0.00
1000	1.08 ±0.19	0.04 ±0.02	1.00 ±0.20	0.04 ±0.02	1.37 ±0.34	97.02 ±0.25	96.87 ±0.01	92.90 ±0.79	96.88 ±0.00	93.75 ±0.00

Table A.8: **Unit-Ball**. Effect of variation of N_q, the maximum number of samples the algorithm can use for training, without using a UCB (i.e., $C_1 = 0$) on error estimates. We keep validation data size N_v fixed at 4000 and use error threshold $\epsilon_a = 1\%$. We report the mean and std. deviation over 10 runs with different random seeds.

N_q	Error (%)					Coverage (%)				
	TBAL	**AL+SC**	**PL+SC**	**AL**	**PL**	**TBAL**	**AL+SC**	**PL+SC**	**AL**	**PL**
100	0.78 ±0.19	0.29 ±0.25	0.24 ±0.15	16.93 ±2.48	12.53 ±2.24	71.40 ±5.64	19.31 ±8.77	35.90 ±10.66	99.69 ±0.00	99.38 ±0.00
200	0.49 ±0.13	0.16 ±0.09	0.17 ±0.08	7.85 ±1.69	7.08 ±1.82	95.28 ±1.29	59.41 ±10.32	63.77 ±9.15	99.38 ±0.00	98.75 ±0.00
400	0.40 ±0.12	0.15 ±0.07	0.15 ±0.08	1.81 ±0.58	3.25 ±0.81	97.63 ±0.24	92.06 ±2.63	83.64 ±4.76	98.75 ±0.00	97.50 ±0.00
600	0.34 ±0.13	0.12 ±0.05	0.17 ±0.08	0.44 ±0.14	2.25 ±0.56	97.36 ±0.20	97.07 ±0.57	87.92 ±2.68	98.12 ±0.00	96.25 ±0.00
800	0.28 ±0.10	0.07 ±0.03	0.15 ±0.06	0.12 ±0.05	1.76 ±0.50	97.10 ±0.23	97.36 ±0.11	88.96 ±2.14	97.50 ±0.00	95.00 ±0.00
1000	0.25 ±0.10	0.03 ±0.02	0.19 ±0.08	0.04 ±0.02	1.37 ±0.34	96.90 ±0.21	96.84 ±0.03	89.61 ±1.49	96.88 ±0.00	93.75 ±0.00

Table A.9: **Unit-Ball**. Effect of variation of N_q, the maximum number of samples the algorithm can use for training, using a UCB (i.e., $C_1 = 0.25$) on error estimates. We keep validation data size N_v fixed at 4000 and use error threshold $\epsilon_a = 1\%$. We report the mean and std. deviation over 10 runs with different random seeds.

N_v	Error (%)					Coverage (%)				
	TBAL	**AL+SC**	**PL+SC**	**AL**	**PL**	**TBAL**	**AL+SC**	**PL+SC**	**AL**	**PL**
2000	0.0 ±0.0	0.0 ±0.0	0.0 ±0.0	0.0 ±0.0	0.0 ±0.0	0.0 ±0.0	0.0 ±0.0	0.0 ±0.0	0.0 ±0.0	0.0 ±0.0
4000	13.88 ±5.42	13.31 ±10.79	12.55 ±7.23	34.34 ±0.32	31.46 ±0.20	0.72 ±0.55	0.48 ±0.04	0.69 ±0.39	95.00 ±0.00	90.00 ±0.00
6000	14.18 ±0.76	11.52 ±0.82	12.21 ±1.49	34.42 ±0.34	31.46 ±0.20	17.29 ±0.72	8.18 ±1.12	11.81 ±2.70	95.00 ±0.00	90.00 ±0.00
8000	13.97 ±0.14	11.31 ±0.51	12.22 ±0.61	34.42 ±0.34	31.46 ±0.20	36.36 ±1.78	23.40 ±1.15	29.99 ±0.97	95.00 ±0.00	90.00 ±0.00
10000	13.42 ±0.29	11.14 ±0.54	12.12 ±0.40	34.42 ±0.34	31.46 ±0.20	43.79 ±0.93	33.38 ±0.72	39.40 ±0.50	95.00 ±0.00	90.00 ±0.00

Table A.10: **Tiny-ImageNet**. Effect of variation of validation data size (N_v), without using a UCB (i.e., $C_1 = 0$) on error estimates. We keep training data size N_q fixed at $10K$ and use error threshold $\epsilon_a = 10\%$. We report the mean and std. deviation over 10 runs with different random seeds.

N_v	Error (%)					Coverage (%)				
	TBAL	**AL+SC**	**PL+SC**	**AL**	**PL**	**TBAL**	**AL+SC**	**PL+SC**	**AL**	**PL**
2000	0.0 ±0.0	0.0 ±0.0	0.0 ±0.0	0.0 ±0.0	0.0 ±0.0	0.0 ±0.0	0.0 ±0.0	0.0 ±0.0	0.0 ±0.0	0.0 ±0.0
4000	10.50 ±6.01	7.37 ±4.57	6.04 ±1.85	34.20 ±0.32	31.52 ±0.27	0.47 ±0.05	0.48 ±0.06	0.43 ±0.01	95.00 ±0.00	90.00 ±0.00
6000	10.61 ±0.62	7.71 ±1.03	8.53 ±1.70	34.42 ±0.34	31.46 ±0.20	10.16 ±1.10	4.31 ±1.10	7.13 ±1.18	95.00 ±0.00	90.00 ±0.00
8000	9.90 ±0.63	6.80 ±0.77	7.81 ±0.85	34.42 ±0.34	31.46 ±0.20	25.84 ±1.57	14.43 ±2.01	19.23 ±1.43	95.00 ±0.00	90.00 ±0.00
10000	8.97 ±0.36	6.87 ±0.48	7.32 ±0.49	34.42 ±0.34	31.46 ±0.20	32.19 ±1.34	21.96 ±1.35	27.01 ±0.98	95.00 ±0.00	90.00 ±0.00

Table A.11: **Tiny-ImageNet**. Effect of variation of validation data size (N_v), using a UCB (i.e., $C_1 = 0.25$) on error estimates. We keep training data size N_q fixed at $10K$ and use error threshold $\epsilon_a = 10\%$. We report the mean and std. deviation over 10 runs with different random seeds.

N_q	Error (%)					Coverage (%)				
	TBAL	**AL+SC**	**PL+SC**	**AL**	**PL**	**TBAL**	**AL+SC**	**PL+SC**	**AL**	**PL**
2000	14.02 ±0.26	11.49 ±0.80	12.17 ±0.35	52.34 ±1.16	42.94 ±0.26	24.34 ±0.86	14.41 ±1.00	25.13 ±0.58	99.00 ±0.00	98.00 ±0.00
4000	14.10 ±0.77	11.58 ±0.26	11.92 ±0.39	43.14 ±0.33	36.07 ±0.41	34.16 ±1.00	21.84 ±1.36	33.41 ±0.65	98.00 ±0.00	96.00 ±0.00
6000	13.55 ±0.17	11.33 ±0.35	12.31 ±0.16	38.73 ±0.59	33.51 ±0.19	37.80 ±1.05	28.59 ±1.53	38.14 ±0.85	97.00 ±0.00	94.00 ±0.00
8000	13.79 ±0.27	11.72 ±0.32	12.36 ±0.30	36.06 ±0.30	32.33 ±0.32	42.00 ±1.71	32.00 ±1.12	39.64 ±1.07	96.00 ±0.00	92.00 ±0.00
10000	13.26 ±0.35	11.42 ±0.28	12.14 ±0.45	34.27 ±0.21	31.47 ±0.17	43.63 ±0.38	33.80 ±0.82	39.23 ±0.37	95.00 ±0.00	90.00 ±0.00

Table A.12: **Tiny-ImageNet**. Effect of variation of N_q, the maximum number of samples the algorithm can use for training, without using a UCB (i.e., $C_1 = 0$) on error estimates. We keep validation data size N_v fixed at $10K$ and use error threshold $\epsilon_a = 10\%$. We report the mean and std. deviation over 5 runs with different random seeds.

N_q	Error (%)					Coverage (%)				
	TBAL	**AL+SC**	**PL+SC**	**AL**	**PL**	**TBAL**	**AL+SC**	**PL+SC**	**AL**	**PL**
2000	9.22 ±1.04	7.42 ±0.71	7.48 ±0.32	52.34 ±1.16	42.94 ±0.26	17.51 ±1.16	9.33 ±0.66	17.02 ±1.32	99.00 ±0.00	98.00 ±0.00
4000	9.30 ±0.38	6.97 ±0.39	7.37 ±0.21	43.14 ±0.33	36.07 ±0.41	25.01 ±1.20	14.25 ±1.71	22.29 ±0.61	98.00 ±0.00	96.00 ±0.00
6000	9.12 ±0.22	6.85 ±0.26	7.49 ±0.35	38.73 ±0.59	33.51 ±0.19	28.06 ±0.75	17.51 ±0.36	25.60 ±0.34	97.00 ±0.00	94.00 ±0.00
8000	9.21 ±0.14	7.38 ±0.53	7.71 ±0.25	36.06 ±0.30	32.33 ±0.32	30.88 ±0.64	21.18 ±0.90	27.26 ±0.78	96.00 ±0.00	92.00 ±0.00
10000	8.95 ±0.23	7.10 ±0.26	7.42 ±0.36	34.27 ±0.21	31.47 ±0.17	32.31 ±1.21	22.34 ±0.61	27.36 ±0.59	95.00 ±0.00	90.00 ±0.00

Table A.13: **Tiny-ImageNet**. Effect of variation of N_q, the maximum number of samples the algorithm can use for training, using a UCB (i.e., $C_1 = 0.25$) on error estimates. We keep validation data size N_v fixed at $10K$ and use error threshold $\epsilon_a = 10\%$. We report the mean and std. deviation over 5 runs with different random seeds.

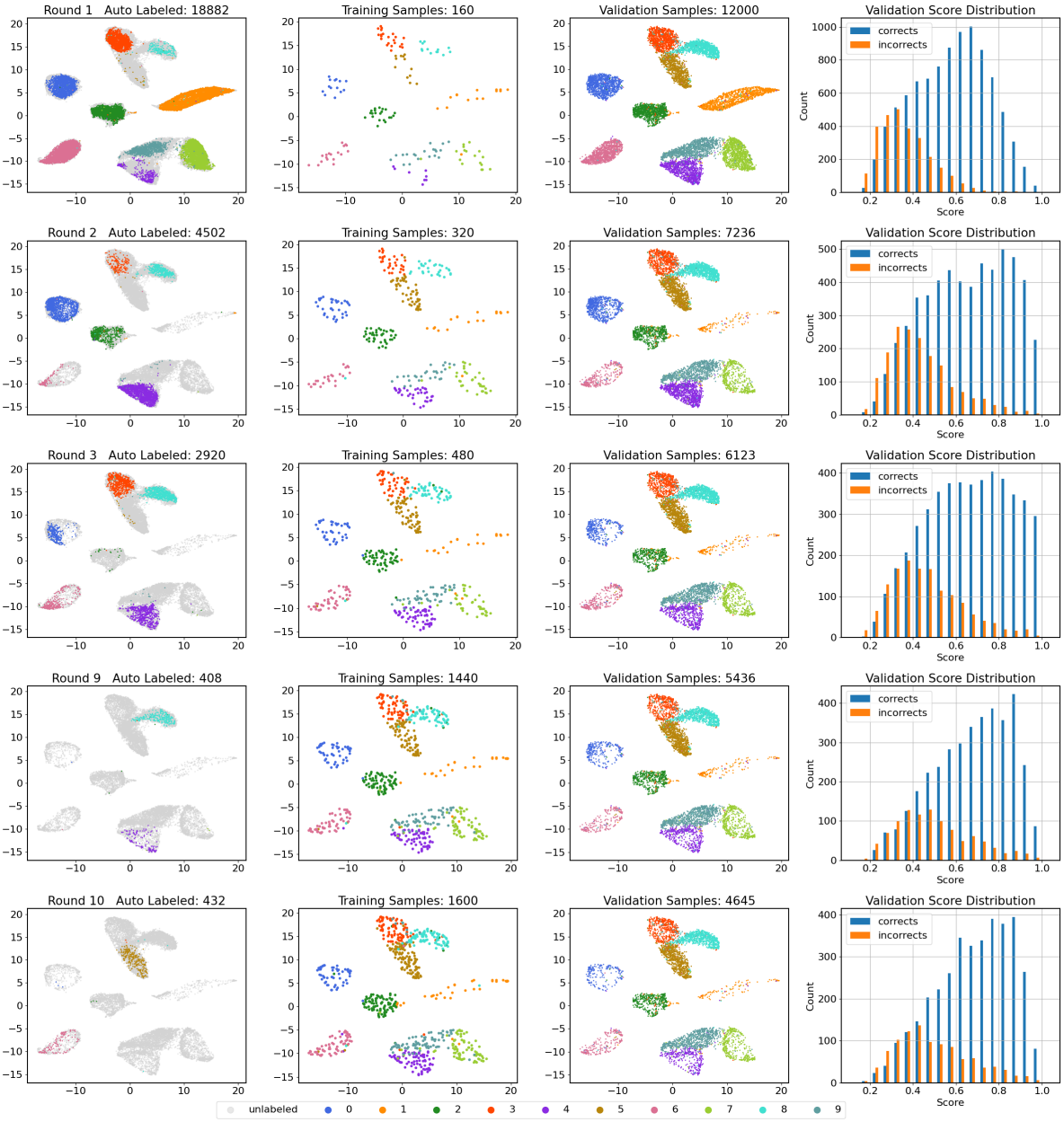


Figure A.5: Auto-labeling MNIST data using linear classifiers. Validation size = 12k. Maximum training samples = 1600. Each round algorithm queries 160 samples. Coverage of auto-labeling is 62.9% with 98.0% accuracy. For the rounds we show, the test error rates are 21.4%, 13.9%, 12.5%, 10.2%, and 9.8%, respectively. For four columns (left to right), we show: **a**) The samples that are labeled by TBAL in this round. **b**) The embeddings for training samples. **c**) The embeddings for validation data points. **d**) The score distribution for the validation dataset.

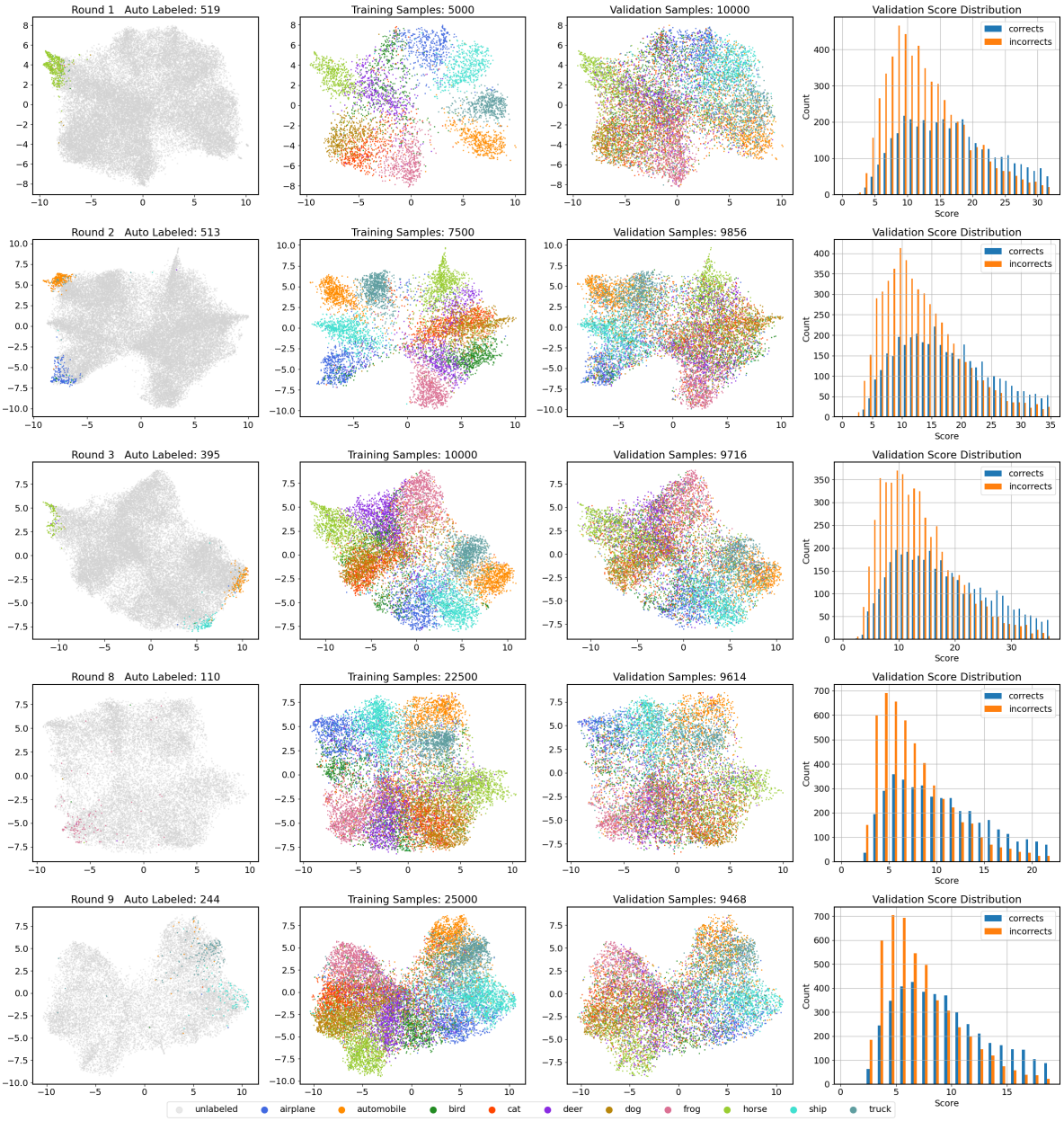


Figure A.6: Auto-labeling CIFAR-10 data using a small network and energy scores. Validation size = 10k. Maximum training samples = 25k. Each round algorithm queries 2500 samples. Coverage of auto-labeling is 5.3% with 90.0% accuracy. For the rounds we show, the test error rates are 56.6%, 55.2%, 55.6%, 53.0%, and 49.3%, respectively. For four columns (left to right), we show: **a**) The samples that are labeled by TBAL in this round. **b**) The embeddings for training samples. **c**) The embeddings for validation data points. **d**) The score distribution for the validation dataset.

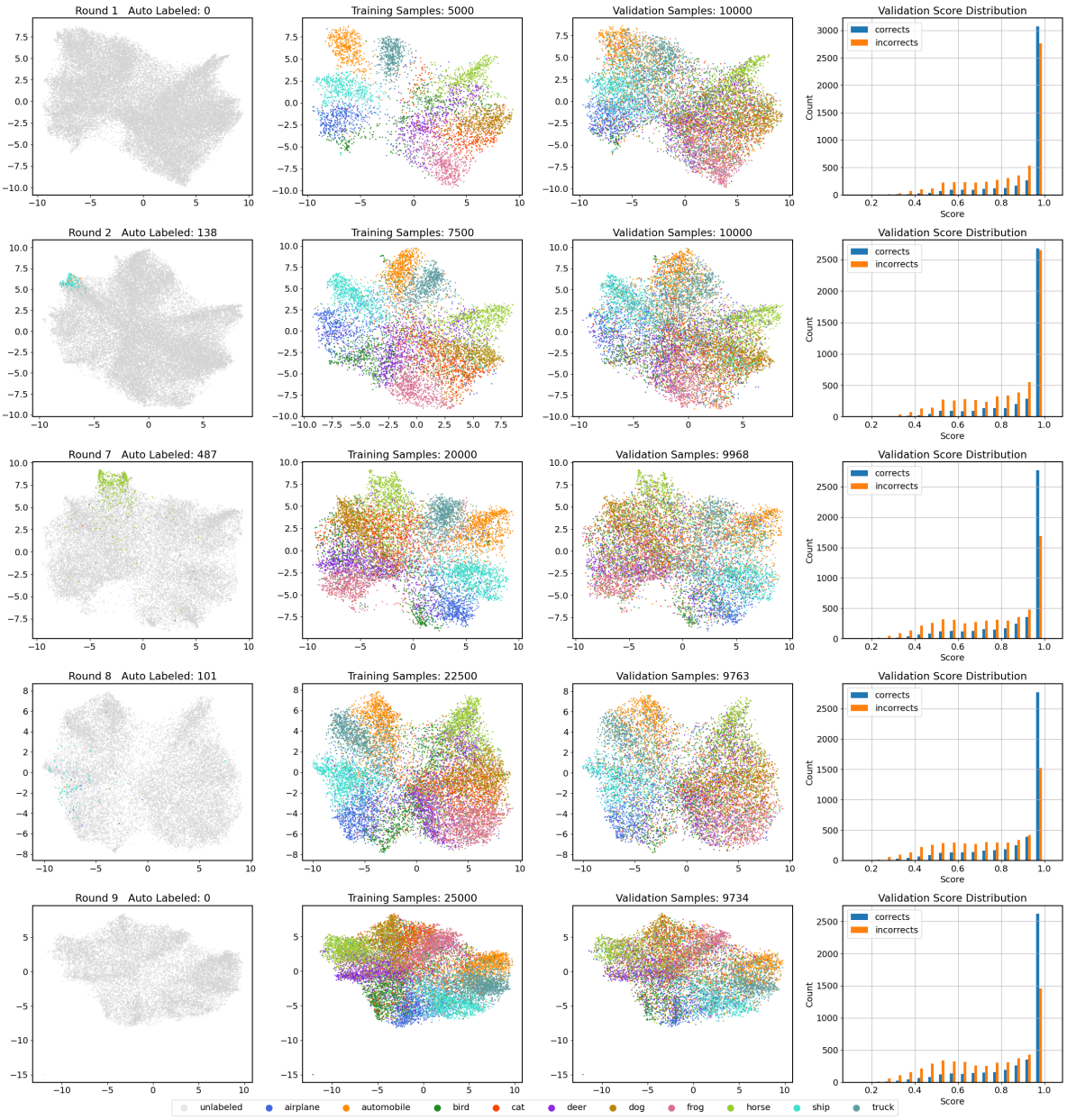


Figure A.7: Auto-labeling CIFAR-10 data using a small network and softmax scores. Validation size = 10k. Maximum training samples = 25k. Each round of the algorithm queries 2500 samples. Coverage of auto-labeling is 2.3% with 91.0% accuracy. For the rounds visualized here in each row, the test error rates of the trained classifiers are 56.6%, 59.1%, 52.8%, 50.5%, and 51.7% respectively. For four columns (left to right), we show: **a**) The samples that are labeled by TBAL in this round. **b**) The embeddings for training samples. **c**) The embeddings for validation data points. **d**) The score distribution for the validation dataset.

Appendix B

Appendix to Chapter 4

The Appendix is organized as follows. We provide deferred details of the background and motivation section in Appendix B.1 of the method in Appendix B.2. Then, in Appendix B.3, we provide additional experimental results and details of the experiment protocol and hyperparameters used for the experiments.

B.1 Appendix to the Background and Motivation Section

B.1.1 Detailed comparison with active learning and self-training

To illustrate the differences between TBAL and the combination of active learning (AL) and self-training for the task of data labeling, we run an experiment on the 2 concentric circles data setting as used in Vishwakarma et al. (2023). The details are as follows:

Data setting. We generate two concentric circles with points in the outer circle belonging to one class and the inner circle belonging to the other class. The total number of points generated is 10,000, of which we use 2000 for validation.

Methods. We run TBAL, AL+Self-Training, and AL+Self-Training+SC, using logistic regression. The combination of AL+Self-Training means, in each iteration, the algorithm queries human-labeled data points and pseudo-labels the points in the unlabeled data using self-training and

adds both the human-labeled and pseudo-labeled points to the training pool. With this procedure, AL+Self-Training first learns the best classifier ($\hat{h}_{\text{al}-\text{st}}$) with the given budget of maximum training points (N_t) that can be queried from humans. Then it auto-labels all the remaining unlabeled points with this classifier's predictions. For AL+Self-Training+SC, we do selective auto-labeling using $\hat{h}_{\text{al}-\text{st}}$, i.e., only auto-label the points where the classifier will have an error at most ϵ_a. We use ϵ_a =1% here.

Results and discussion. The Figure B.1 shows auto-labeling error and coverage achieved by these methods when run with different choices of human-labeled data budget for training. First, we can see that even with linear classifiers TBAL is able to auto-label a huge chunk of the data (high coverage) while maintaining auto-labeling error below the tolerance level of 1% On the other hand, methods like AL+Self-Training (+SC) that try to first learn the optimal classifier in the given function class either have high auto-labeling error or very low coverage. These results are also consistent with the observations in Vishwakarma et al. (2023) on the comparison between TBAL and AL, AL+SC. While such findings confirm the notion that there are differences—and, at least in some settings, advantages—for the TBAL approach compared to other techniques, we reiterate that our goal is to understand and improve the role of the confidence function within TBAL, rather than comparing TBAL to other techniques.

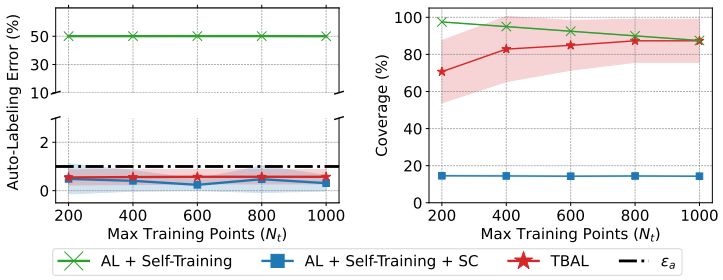
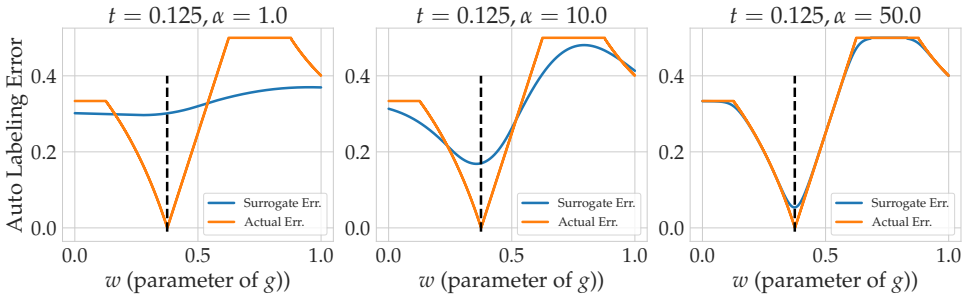


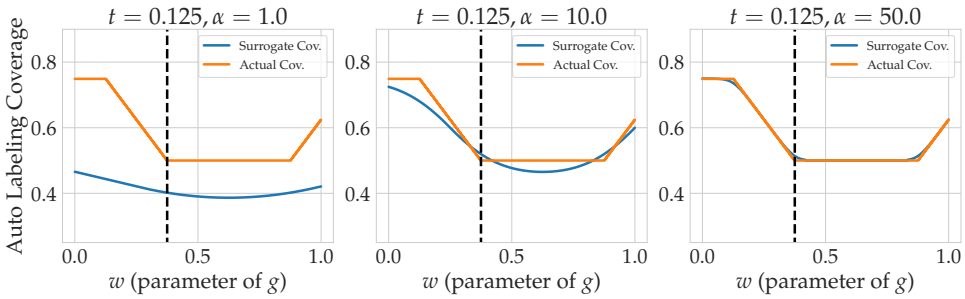
Figure B.1: Results of experiment on 2-concentric circles to show the differences between TBAL, AL, and ST.

B.1.2 Details of the motivating experiment in section 4.2

We run TBAL for a single round on the CIFAR-10 dataset with a SimpleCNN classification model with around 5.8M parameters (Hussain, 2021). We randomly sampled 4,000 points for training the classifier and randomly sampled 1,000 points as validation data. We train the model



(a) Auto-labeling error and surrogate error at various α.



(b) Auto-labeling coverage and surrogate coverage at various α.

Figure B.2: Illustration of the tightness of surrogate error and coverage functions based on the choice of α.

to zero training error using minibatch SGD with learning rate 1e-3, weight decay 1e-3 Hanson and Pratt (1988); Krogh and Hertz (1991), momentum 0.9, and batch size 32. The trained model has validation accuracy around 55%, implying we could hope to get coverage around 55%. We run the auto-labeling procedure with an error tolerance of 5%.

B.2 Additional Details on the Method

B.2.1 Detailed algorithms

See Algorithms 5, 6 and 7.

B.2.2 Tightness of surrogates.

The surrogate auto-labeling error and coverage introduced to relax the optimization problem (P4.2) is indeed a good approximation of the actual auto-labeling error and coverage. To see this, we use a toy data setting of $x \sim \text{Uniform}(0, 1)$ with $1-$dimensional threshold classifier $h_\theta(x) = \mathbb{1}(x \geq \theta)$. For any x, let true labels $y = h_{0.5}(x)$ and consider the confidence function

$g_w(x) = |w - x|$. Let $\hat{y} = \hbar_{0.25}(x)$ and consider the points on the side where $\hat{y} = 1$. We plot actual and surrogate errors in Figure B.2(a) and the surrogate and actual coverage in Figure B.2(a). for three choices of α. As expected, the gap between the surrogates and the actual functions diminishes as we increase the α.

B.2.3 Active querying strategy.

We employ the margin-random query approach to select the next batch of training data. This method involves sorting points based on their margin (uncertainty) scores and selecting the top Cn_b points, from which n_b points are randomly chosen. This strategy provides a straightforward and computationally efficient way to balance the exploration-exploitation trade-off. It's important to acknowledge the existence of alternative active-querying strategies; however, we adopt the margin-random approach as our standard to maintain a focus on evaluating various choices of confidence functions for auto-labeling. Note that while we use the new confidence scores computed using post-hoc methods for auto-labeling, we do not use these scores in active querying. Instead, we use the softmax scores from the model for this. We do this to avoid conflating the study with the study of active querying strategies. We use $C = 2$ for all experiments.

B.2.4 Glossary

The notation is summarized in Table B.1 below.

Algorithm 5: Threshold-based Auto-Labeling (TBAL)

Input: Unlabeled data X_u, labeled validation data D_{val}, auto labeling error tolerance ϵ_a, N_t training data query budget, seed data size n_s, batch size for active query n_b, calibration data fraction ν, space of thresholds T, coverage lower bound ρ_0, label space $\mathcal{Y}$.

Output: Auto-labeled dataset D_{out}.

procedure TBAL($X_u, D_{\text{val}}, \epsilon_a, N_t, n_s, n_b, \nu, \rho_0, T, \mathcal{Y}$)
  ▷ /*** Initialization. ***/
  $D^{(1)}_{\text{query}} \leftarrow \text{RandomQuery}(X_u, n_s)$. ▷ Randomly select n_s points and get human labels.
  $X^{(1)}_u \leftarrow X_u \setminus \{\mathbf{x} : (\mathbf{x}, y) \in D^{(1)}_{\text{query}}\}$. ▷Remove these points from the unlabeled pool.
  $D^{(1)}_{\text{val}} \leftarrow D_{\text{val}}; D^{(0)}_{\text{train}} \leftarrow \emptyset$. ▷Validation data for the first round is full D_{val}.
  $D_{\text{out}} \leftarrow D^{(1)}_{\text{query}}; n^{(1)}_t \leftarrow n_s; i \leftarrow 1$. ▷Add human-labeled data to the output D_{out}.
  ▷ /*** Run the auto-labeling loop. ***/
  ▷ /* Until no more unlabeled points are left or the budget for training data is over. */
  while $X^{(i)}_u \neq \emptyset$ and $n^{(i)}_t \leq N_t$ **do**
    $D^{(i)}_{\text{train}} \leftarrow D^{(i-1)}_{\text{train}} \cup D^{(i)}_{\text{query}}$. ▷Include human-labeled points in the training data.
    $\hat{h}_i \leftarrow \text{TrainModel}(\mathcal{H}, D^{(i)}_{\text{train}})$. ▷Train a classification model.
    $D^{(i)}_{\text{cal}}, D^{(i)}_{\text{th}} \leftarrow \text{RandomSplit}(D^{(i)}_{\text{val}}, \nu)$. ▷Randomly split current validation data.
    ▷ /*** `Colander` block, to learn the new confidence function $\hat{g}_i$. ***/
    $\hat{g}_i, \hat{\mathbf{t}}'_i \leftarrow \arg\min_{g \in \mathcal{G}, \mathbf{t} \in T^k} -\widetilde{\mathcal{P}}(g, \mathbf{t} \mid \hat{h}_i, D^{(i)}_{\text{cal}}) + \lambda \widetilde{\mathcal{E}}(g, \mathbf{t} \mid \hat{h}_i, D^{(i)}_{\text{cal}})$. ▷ `Colander`.
    ▷ /*** Estimate auto-labeling thresholds using $\hat{g}_i$ and $D^{(i)}_{\text{th}}$. See Algorithm 6. ***/
    $\hat{\mathbf{t}}_i \leftarrow \text{EstThreshold}(\hat{g}_i, \hat{h}_i, D^{(i)}_{\text{th}}, \epsilon_a, \rho_0, T, \mathcal{Y})$.
    ▷ /*** Auto-label the points having scores above the thresholds. ***/
    $\widetilde{D}^{(i)}_u \leftarrow \{(\mathbf{x}, \hat{h}_i(\mathbf{x})) : \mathbf{x} \in X^{(i)}_u\}$.
    $D^{(i)}_{\text{auto}} \leftarrow \{(\mathbf{x}, \hat{y}) \in \tilde{D}^{(i)}_u : \hat{g}_i(\mathbf{x})[\hat{y}] \geq \hat{\mathbf{t}}_i[\hat{y}]\}$.
    $X^{(i)}_u \leftarrow X^{(i)}_u \setminus \{\mathbf{x} : (\mathbf{x}, \hat{y}) \in D^{(i)}_{\text{auto}}\}$. ▷Remove auto-labeled points from unlabeled set.
    $\widetilde{D}^{(i)}_{\text{val}} \leftarrow \{(\mathbf{x}, \hat{h}_i(\mathbf{x})) : (\mathbf{x}, y) \in D^{(i)}_{\text{val}}\}$.
    $D^{(i+1)}_{\text{val}} \leftarrow \{(\mathbf{x}, \hat{y}) \in \tilde{D}^{(i)}_{\text{val}} : \hat{g}_i(\mathbf{x})[\hat{y}] < \hat{\mathbf{t}}_i[\hat{y}]\}$. ▷Remove validation points from the auto-labeling region.
    ▷ /*** Get the next batch of manually labeled data using an active querying. ***/
    $D^{(i+1)}_{\text{query}} \leftarrow \text{ActiveQuery}(\hat{h}_i, X^{(i)}_u, n_b)$.
    $X^{(i+1)}_u \leftarrow X^{(i)}_u \setminus \{\mathbf{x} : (\mathbf{x}, y) \in D^{(i+1)}_{\text{query}}\}$. ▷Remove human-labeled data from the unlabeled pool.
    $D_{\text{out}} \leftarrow D_{\text{out}} \cup D^{(i)}_{\text{auto}} \cup D^{(i+1)}_{\text{query}}$. ▷Add the auto-labeled and manually labeled points in the output data.
    $n^{(i+1)}_t \leftarrow n^{(i)}_t + n_b$.
    $i \leftarrow i + 1$.
  end while
  return D_{out}.
end procedure

Algorithm 6: Estimate Auto-Labeling Threshold

Input: Confidence function $\hat{g}_i$, classifier $\hat{h}_i$, Part of validation data $D_{\text{th}}^{(i)}$ for threshold estimation, auto labeling error tolerance ϵ_a, space of thresholds T, coverage lower bound ρ_0, label space $\mathcal{Y}$.
Output: Auto-labeling thresholds $\hat{\mathbf{t}}_i$, where $\hat{\mathbf{t}}_i[y]$ is the threshold for class y.

procedure EstThreshold($\hat{g}_i, \hat{h}_i, D_{\text{th}}^{(i)}, \epsilon_a, \rho_0, T, \mathcal{Y}$) /*** Estimate thresholds for each class. ***/
  for $y \in \mathcal{Y}$ **do**
    $D_{\text{th}}^{(i,y)} \leftarrow \{(\mathbf{x}', y') \in D_{\text{th}}^{(i)} : y' = y\}$. ▷Group points class-wise.
    ▷ /*** Only evaluate thresholds with est. coverage at least ρ_0. ***/
    $T'_y \leftarrow \{t \in T : \widehat{\mathcal{P}}(\hat{g}_i, t \mid \hat{h}_i, D_{\text{th}}^{(i,y)}) \geq \rho_0\} \cup \{\infty\}$.
    ▷ /*** Estimate auto-labeling error at each threshold. Pick the smallest threshold with the sum of estimated error and C_1 times the std. deviation is below ϵ_a. C_1 is set to 0.25 here. ***/
    $\hat{\mathbf{t}}_i[y] \leftarrow \min\{t \in T'_y : \widehat{\mathcal{E}}_y(\hat{g}_i, t|\hat{h}_i, D_{\text{th}}^{(i,y)}) + C_1\widehat{\zeta}(\widehat{\mathcal{E}}_y(\hat{g}_i, t|\hat{h}_i, D_{\text{th}}^{(i,y)})) \leq \epsilon_a\}$.
  end for
  return $\hat{\mathbf{t}}_i$.
end procedure

Algorithm 7: Active Querying Strategy to Acquire Human-labeled Samples for Training

Input: Classifier $\hat{h}_i$, unlabeled data $X_u^{(i)}$, batch size n_b, constant $C \geq 1$.
Output: $D_{\text{query}}^{(i+1)}$, a subset of $X_u^{(i)}$ of size at most n_b with human (groundtruth) labels.

procedure ActiveQuery($\hat{h}_i, X_u^{(i)}, n_b$)
  $S_u^{(i)} \leftarrow$ Softmax scores from $\hat{h}_i$ for all points in $X_u^{(i)}$.
  $X' \leftarrow$ Top $C \times n_b$ points from $X_u^{(i)}$ sorted in ascending order on the scores $S_u^{(i)}$.
  $D_{\text{query}}^{(i+1)} \leftarrow$ RandomQuery(X', n_b).
  return $D_{\text{query}}^{(i+1)}$.
end procedure

Algorithm 8: Select a Subset of Points Randomly and Obtain Human Labels

Input: X, n.
Output: D, a subset of X of size at most n with human (groundtruth) labels.

procedure RandomQuery(X, n)
  if then$|X| > n$
    $X'' \leftarrow$ randomly select n points from X.
  else
    $X'' \leftarrow X$.
  end if
  $D \leftarrow \{(\mathbf{x}, \texttt{human_label}(\mathbf{x}) : \mathbf{x} \in X''\}$.
  return D.
end procedure

Symbol	Definition
$\mathbb{1}(E)$	indicator function of event E. It is 1 if E happens and 0 otherwise.
$\mathcal{X}$	feature space.
$\mathcal{Y}$	label space i.e. $1, 2, \dots k$.
$\mathcal{H}$	hypothesis space (model class for the classifiers).
$\mathcal{G}$	class of confidence functions.
k	number of classes.
$\mathbf{x}, y$	$\mathbf{x}$ is an element in $\mathcal{X}$ and y is its true label.
f^*	unknown groundtruth labeling function.
h	a hypothesis (model) in $\mathcal{H}$.
g	confidence function $g : \mathcal{X} \to \Delta^k$.
ϵ_a	auto-labeling error tolerance.
X_u	given pool of unlabeled data points.
$X_u^{(i)}$	unlabeled data left at the beginning of ith round.
A	indices of points that are auto-labeled.
$X_u(A)$	subset of points in X_u with indices in A, i.e., auto-labeled points.
$\hat{h}^{(i)}$	ERM solution and auto-labeling thresholds respectively in ith round.
$\mathbf{t}$	k dimensional vector of thresholds.
$\mathbf{t}[y]$	yth entry of $\mathbf{t}$ i.e. the threshold for class y.
$g(\mathbf{x})[y]$	the confidence score for class y output by g on data point $\mathbf{x}$.
N_u	number of unlabeled points, i.e., size of X_u.
N_t	number of manually labeled points that can be used for training h.
N_a	Total auto-labeled points in D_{out}.
ν	fraction of D_{val} that can be used for training post-hoc calibrator.
$\tilde{y}_i$	label assigned to the ith point. It could be either y_i or $\hat{y}_i$.
y_i	groundtruth label for the ith point.
$\hat{y}_i$	predicted label for the ith point by classifier.
$D_{\text{query}}^{(i)}$	labeled data queried from oracle (human) in the ith round.
$D_{\text{train}}^{(i)}$	training data to learn $\hat{h}^{(i)}$ in the ith round.
$D_{\text{val}}^{(i)}$	validation data in the ith round.
$D_{\text{cal}}^{(i)}$	calibration data in the ith round to learn a post-hoc g.
$D_{\text{th}}^{(i)}$	part of validation data in the ith round to estimate threshold $\mathbf{t}$.
$D_{\text{auto}}^{(i)}$	part of $X_u^{(i)}$ that got auto-labeled in the ith round.
D_{out}	Output labeled data, including auto-labeled and human-labeled data.
$\mathcal{E}(g, \mathbf{t} \mid h)$	population level auto-labeling error, see eq. (4.2).
$\mathcal{P}(g, \mathbf{t} \mid h)$	population level auto-labeling coverage, see eq. (4.2).
$\widehat{\mathcal{E}}(g, \mathbf{t} \mid h, D)$	estimated auto-labeling error, see eq. (4.4).
$\widehat{\mathcal{P}}(g, \mathbf{t} \mid h, D)$	estimated auto-labeling coverage, see eq. (4.3).
$\widetilde{\mathcal{E}}(g, \mathbf{t} \mid h, D)$	surrogate estimated auto-labeling error, see eq. (4.6).
$\widetilde{\mathcal{P}}(g, \mathbf{t} \mid h, D)$	surrogate estimated auto-labeling coverage, see eq. (4.5).

Table B.1: Glossary of variables and symbols used in this paper.

B.3 Additional Experiments and Details

B.3.1 Train-time and post-hoc methods

Train-time methods

1. *Vanilla*: Neural networks are commonly trained by minimizing the cross entropy loss using stochastic gradient descent (SGD) with momentum (Amari, 1993; Bottou, 2012). We refer to this as the Vanilla training method. We also include weight decay to mitigate the overconfidence issue associated with this method (Guo et al., 2017).

2. *Squentropy* (Hui et al., 2023): This method adds the average square loss over the incorrect classes to the cross-entropy loss. This simple modification to the Vanilla method leads to the end model with better test accuracy and calibration.

3. *Correctness Ranking Loss (CRL)* (Moon et al., 2020): This method includes a term in the loss function of the vanilla training method so that the confidence scores of the model are aligned with the ordinal rankings criterion (Hendrycks and Gimpel, 2017a; Corbière et al., 2019). The confidence functions satisfying this criterion produce high scores on points where the probability of correctness is high and low scores on points with low probabilities of being correct.

4. *FMFP* (Zhu et al., 2022) aims to align confidence scores with the ordinal rankings criterion. It uses Sharpness Aware Minimizer (SAM) (Foret et al., 2021) to train the model, with the expectation that the flat minima would benefit the ordinal rankings objective of the confidence function.

Post-hoc methods

1. *Temperature scaling* (Guo et al., 2017): This is a variant of Platt scaling (Guo et al., 2017), a classic and one of the easiest parametric methods for post-hoc calibration. It rescales the logits by a learnable scalar parameter and has been shown to work well for neural networks.

2. *Top-Label Histogram-Binning* (Gupta and Ramdas, 2022): Since TBAL assigns the top

labels (predicted labels) to the selected unlabeled points, it is appealing to only calibrate the scores of the predicted label. Building upon a rich line of histogram-binning methods (non-parametric) for post-hoc calibration (Zadrozny and Elkan, 2002), this method focuses on calibrating the scores of predicted labels.

3. *Scaling-Binning* (Kumar et al., 2019): This method combines parametric and non-parametric methods. It first applies temperature scaling and then bins the confidence function values to ensure calibration.

4. *Dirichlet Calibration* (Kull et al., 2019): This method models the distribution of predicted probability vectors separately on instances of each class and assumes the class conditional distributions are Dirichlet distributions with different parameters. It uses linear parameterization for the distributions, which allows easy implementation in neural networks as additional layers and softmax output.

Note: For binning methods, uniform mass binning (Zadrozny and Elkan, 2002) has been a better choice over uniform width binning. Hence, we use uniform mass binning as well.

B.3.2 Detailed dataset and model

1. The MNIST dataset LeCun (1998) consists of 28×28 grayscale images of hand-written digits across 10 classes. It was used alongside the LeNet5 LeCun et al. (1998), a convolutional neural network, for auto-labeling.

2. The CIFAR-10 dataset Krizhevsky et al. (2009a) contains $3 \times 32 \times 32$ color images across 10 classes. We utilized its raw pixel matrix in conjunction with SimpleCNN Hussain (2021), a convolutional neural network with approximately 5.8M parameters, for auto-labeling.

3. Tiny-ImageNet Le and Yang (2015b) is a color image dataset that consists of 100K images across 200 classes. Instead of using the $3 \times 64 \times 64$ raw pixel matrices as input, we utilized CLIP Radford et al. (2021) to derive embeddings within the $\mathbb{R}^{512}$ vector space. We used a 3-layer perceptron (1,000-500-300) as the auto-labeling model.

Method	Hyperparameter	Values
Common	optimizer	SGD
	learning rate	0.001, 0.01, 0.1
	batch size	32, 256
	max epoch	50, 100
	weight decay	0.001, 0.01, 0.1
	momentum	0.9
CRL	rank target	softmax
	rank weight	0.7, 0.8, 0.9
FMFP	optimizer	SAM

Table B.2: Hyperparameters swept over for train-time methods. Those listed next to Common are the hyperparameters for the four train-time methods: Vanilla, CRL, FMFP, and Squentropy. Therefore, we do not list those again for each method. Note that for FMFP, we used SAM optimizer instead of SGD. For each method, we swept through all possible combinations of the possible values for each hyperparameter. Underlined values are only used on TinyImageNet since it is a complicated dataset containing 200 classes.

4. 20 Newsgroups Mitchell (1999); Pedregosa et al. (2011) is a natural language dataset comprising around 18,000 news posts across 20 topics. We used the FlagEmbedding Xiao et al. (2023) to map the textual data into $\mathbb{R}^{1024}$ embeddings. We used a 3-layer perceptron (1,000-500-30) as the auto-labeling model.

B.3.3 Detailed experiments protocol

We predefined TBAL hyperparameters for each dataset-model pair and the hyperparameters we will sweep for each train-time and post-hoc method in Table B.2 and Table B.3, respectively. For a dataset-model pair, initially, we perform a hyperparameter search for the train-time method. Subsequently, we optimize the hyperparameters for post-hoc methods while keeping the train-time method fixed with the previously found optimum hyperparameter for that dataset-model pair.

We fix the hyperparameters for the train-time method while searching hyperparameters for the post-hoc method to alleviate computational budget throttle. We effectively reduce the search space to the sum of the cardinalities of unique hyperparameter combinations across the two methods instead of a larger multiplicative product. Furthermore, due to the independent nature of these hyperparameter combinations, TBAL runs can be highly parallelized to expedite the

Method	Hyperparameter	Values
Temperature scaling	optimizer	Adam
	learning rate	0.001, 0.01, 0.1
	batch size	64
	max epoch	500
	weight decay	0.01, 0.1, 1
Top-label histogram binning	points per bin	25, 50
Scaling-binning	number of bins	15, 25
	learning rate	0.001, 0.01, 0.1
	batch size	64
	max epoch	500
	weight decay	0.01, 0.1, 1
Dirichlet calibration	regularization parameter	0.001, 0.01, 0.1
Ours	λ	10, 100
	features key	concat
	class-wise	independent
	optimizer	Adam
	learning rate	0.01, 0.1
	max epoch	500
	weight decay	0.01, 0.1, 1
	batch size	64
	regularize	false
	α	0.01, 0.1, 1

Table B.3: Hyperparameters swept over for post-hoc methods. For each method, we swept through all possible combinations of the possible values for each hyperparameter.

search process.

Since TBAL operates iteratively to acquire human labels for model training, selecting hyperparameters at each round of TBAL could quickly become intractable and lose its practical significance. To better align with its practical usage, we only conducted a hyperparameter search for the initial TBAL round. The specific set of hyperparameters used for the search are reported in Table B.3.

After completing the hyperparameter search for train-time and post-hoc methods, the determined hyperparameter combinations are subjected to a full evaluation across all iterations of TBAL. At the end of each iteration, the auto-labeled points are evaluated against their ground truth labels to determine their auto-labeling error. These points are then added to the auto-labeled set, where their ratio to the total amount of unlabeled data determines the coverage. This iterative

process continues until all unlabeled data are exhaustively labeled by either the oracle or through auto-labeling in the final iteration. The auto-labeling error and coverage at the final iteration of TBAL are then recorded.

Since TBAL incorporates randomized components as detailed in Algorithm 5, we ran the algorithm 5 times, each with a unique random seed while maintaining the same hyperparameter combination. We then recorded the results from the final iteration of these runs and calculated the mean and standard deviation of both auto-labeling error and coverage. These figures are reported in Table 4.2.

A limitation of the grid search approach in hyperparameter optimization becomes apparent when our predefined hyperparameter choices result in sub-optimal coverage and auto-labeling errors. Using these sub-optimal hyperparameters can adversely affect the multi-round iterative process in TBAL, prompting the need for repetitive searches to find more effective hyperparameters. When encountering such scenarios, TBAL users should explore additional hyperparameter options until satisfactory performance is achieved in the initial round. However, we opted for a more straightforward approach to hyperparameter selection, mindful of the computational demands of repeatedly optimizing multiple hyperparameters across different methods. In scenarios expressed conditionally, we retained the top-1 hyperparameter combination for any given method if it achieved the highest coverage while adhering to the specified error margin (ϵ_a). If no hyperparameter combinations yielded an auto-labeling error at most equal to the error margin (ϵ_a), we then chose the hyperparameter combination with the lowest auto-labeling error, regardless of its coverage. In the case of ties, we resolved them through random selection. This process results in obtaining singular values for each choice of hyperparameter after completing each method's hyperparameter search.

B.3.4 Hyperparameters

The hyperparameters and their values we swept over are listed in Table B.2 and B.3 for train-time and post-hoc methods, respectively.

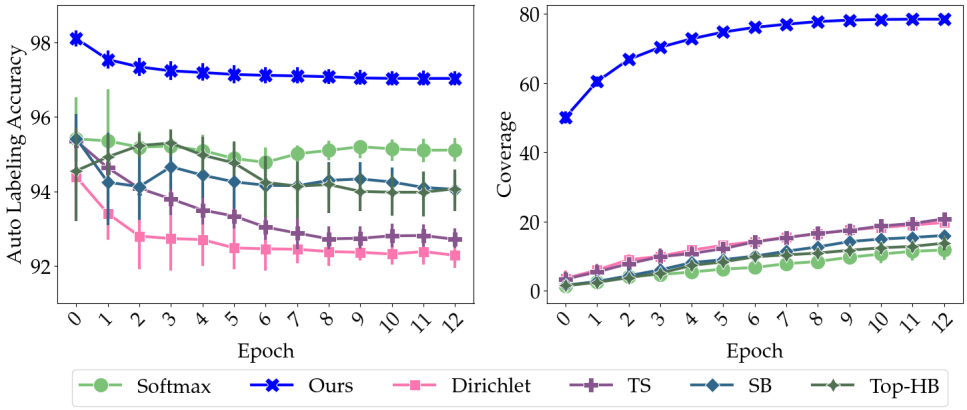


Figure B.3: Per-epoch metrics for all post-hoc methods for CIFAR10 setting. (left) Auto-labeling accuracy (right) Coverage. Train-time method is vanilla.

B.3.5 Compute resources and time

Our experiments were conducted on machines equipped with the NVIDIA RTX A6000 and NVIDIA GeForce RTX 4090 GPUs. The wall clock time of our method is similar to other post-hoc methods. For instance, a single run on the CIFAR-10 setting on NVIDIA RTX A6000 takes around 1.5 hours with post-hoc methods and roughly 1 hour without post-hoc methods. The additional time taken by our method over the baselines not doing any post-hoc calibration is traded-off the by the quality and quantity of the auto-labeled data it outputs. We leave a thorough benchmarking of wall clock time and its optimization for future work.

B.3.6 Single vs multi-round TBAL

We further demonstrate that the performance gains are due to the use of `Colander`, even if methods use multiple rounds. To do so, we show the evolution of coverage and error over multiple rounds in Figure B.3. The effects of using `Colander` are visible from the first round itself, and the following rounds improve performance further. We also run a single round (passive) variant of TBAL where we sample all

Post-hoc	**Err (↓)**	**Cov (↑)**
Softmax	2.7 ± 0.54	11.06 ± 1.46
TS	3.04 ± 0.49	12.03 ± 1.98
Dirichlet	2.98 ± 0.32	11.22 ± 2.1
SB	2.72 ± 0.34	9.75 ± 1.33
Top-HB	1.83 ± 0.61	5.50 ± 1.08
Ours	**2.02 ± 0.28**	**49.62 ± 0.69**

Table B.4: Results with a single round of auto-labeling. Dataset and model: CIFAR-10 setting in the paper.

Feature	Model	Error	Coverage
Pre-logits	Two Layer	4.6 ± 0.3	82.8 ± 0.5
Logits	Two Layer	3.2 ± 1.3	82.8 ± 0.3
Concat	Two Layer	3.3 ± 0.8	82.9 ± 0.4

Table B.5: Auto-labeling error and coverage for the 3 feature representations we could use for 20 Newsgroup. As we can see, the feature representation does not lead to a significant difference in auto-labeling error and coverage.

Feature	Model	Error	Coverage
Pre-logits	Two Layer	2.1 ± 0.5	79.0 ± 0.2
Logits	Two Layer	3.1 ± 0.4	76.5 ± 0.9
Concat	Two Layer	2.3 ± 0.5	79.0 ± 0.3

Table B.6: Auto-labeling error and coverage for the 3 feature representations we could use for CIFAR10 SimpleCNN. As we can see, the feature representation does not lead to a significant difference in auto-labeling error and coverage.

the human-labeled points for training (N_t) randomly at once, train a classifier, do auto-labeling, and then stop. This setting avoids confounding due to multiple rounds. We observe that using `Colander` yields significantly higher coverage in comparison to the baselines (see Table B.4). This reinforces the fact that the gains in the multi-round TBAL are directly due to `Colander`, while multiple rounds of data selection, training, and auto-labeling are superior to this single round version.

B.3.7 Experiments on `Colander` input

Figure 4.4 illustrates that we could use logits (last layer's representations), pre-logits (second last layer's representations), or the concatenation of these two as the input to g. To help us decide which one we should use, we conduct a hyperparameter search for input features on the CIFAR-10 and 20 Newsgroup dataset using the Squentropy train-time method. Table B.5 and B.6 present the auto-labeling error and coverage of using the 3 types of feature representations. As we can see, all feature representation leads to a similar auto-labeling error and coverage, and in some cases, it is better to include pre-logits as well. Thus, we use concatenated representation (Concat), for more flexibility.

Post-hoc	$\epsilon_a = 0.01$		$\epsilon_a = 0.025$		$\epsilon_a = 0.05$		$\epsilon_a = 0.075$		$\epsilon_a = 0.1$	
	Err (↓)	**Cov (↑)**	**Err (↓)**	**Cov (↑)**	**Err (↓)**	**Cov (↑)**	**Err (↓)**	**Cov (↑)**	**Err (↓)**	**Cov (↑)**
Softmax	5.86 ± 0.38	12.73 ± 1.61	5.86 ± 0.38	12.73 ± 1.61	4.78 ± 0.21	14.01 ± 2.08	6.80 ± 0.47	16.73 ± 1.19	9.03 ± 0.17	21.28 ± 0.82
TS	8.19 ± 0.88	19.44 ± 1.16	8.19 ± 0.88	19.44 ± 1.16	7.26 ± 0.29	23.15 ± 0.7	9.24 ± 0.78	22.49 ± 0.74	11.63 ± 0.51	25.79 ± 1.97
Dirichlet	8.22 ± 0.4	16.94 ± 1.2	8.22 ± 0.4	16.94 ± 1.2	7.6 ± 0.48	22.36 ± 1.18	9.68 ± 0.82	18.65 ± 0.97	11.26 ± 1.16	24.91 ± 2.09
SB	6.15 ± 0.52	11.74 ± 0.57	6.15 ± 0.52	11.74 ± 0.57	6.09 ± 0.35	18.58 ± 1.13	7.81 ± 0.65	17.37 ± 1.3	9.13 ± 1.08	20.52 ± 1.11
Top-HB	5.76 ± 0.42	9.89 ± 0.55	5.76 ± 0.42	9.89 ± 0.55	5.95 ± 0.7	15.58 ± 1.92	7.45 ± 0.8	13.84 ± 0.78	8.71 ± 1.37	17.9 ± 0.56
Ours	**1.2 ± 0.18**	**78.33 ± 0.76**	**1.32 ± 0.21**	**78.75 ± 0.4**	**2.96 ± 0.2**	**78.48 ± 0.17**	**4.3 ± 0.23**	**78.94 ± 0.42**	**6.29 ± 0.5**	**78.97 ± 0.46**

Table B.7: ϵ_a **variation.** Dataset: CIFAR-10, Train-time method: Vanilla.

B.3.8 Experiments on ϵ_a

We run TBAL with five values of $\epsilon_a \in \{0.01, 0.025, 0.05, 0.075, 0.1\}$ and report the results in Table B.7. As expected, the auto-labeling error is high with larger values of and smaller with small ϵ_a.

B.3.9 Experiments on different architectures

In TBAL, it is not a priori clear what model the practitioner should use. The overall system is flexible enough to work with any chosen model class. Our focus is on evaluating the effect of various training time and post-hoc methods designed to improve the confidence functions for any given model. To answer the query, we ran experiments with Resnet18 and ViT models in the CIFAR-10 setting (see Table B.8). As we expected, there are variations in the results in the baselines due to model choice,s but our method maintains high performance irrespective of the classification model used. This is due to its ability to learn confidence scores tailored for TBAL.

Post-hoc	Err (↓)	Cov (↑)
Softmax	14.02 ± 1.83	2.03 ± 0.31
TS	19.32 ± 2.51	2.54 ± 0.33
Dirichlet	17.27 ± 3.26	2.87 ± 0.55
SB	9.22 ± 10.91	0.46 ± 0.51
Top-HB	0.00 ± 0.00	0.00 ± 0.00
Ours	**2.62 ± 0.32**	**75.56 ± 0.15**

Post-hoc	Err (↓)	Cov (↑)
Softmax	4.48 ± 0.23	33.24 ± 1.14
TS	6.38 ± 0.47	39.14 ± 1.96
Dirichlet	6.30 ± 0.41	37.99 ± 1.47
SB	5.16 ± 0.23	35.32 ± 1.36
Top-HB	4.46 ± 0.40	29.66 ± 0.74
Ours	**2.85 ± 0.25**	**78.56 ± 0.54**

Table B.8: Model variation. CIFAR-10 dataset with ViT (Left) and ResNet18 (Right), Train-time method Vanilla.

B.3.10 Experiments on N_t, N_v and ν

We need to understand the effect of training data query budget, i.e., N_t, the total validation data N_v, and the data that can be used for calibrating the model, i.e., the calibration data fraction ν, on the auto-labeling objective. As varying these hyperparameters on each train-time method is expensive, we experimented with only Squentropy as it was the best-performing method across settings for various datasets.

When we vary the budget for training data N_t, we observe from Figure B.4 that our method does not require a lot of data to train the base model, i.e., achieving low auto-labeling error and high coverage with a low budget. While other methods benefit from having more training data for auto-labeling objectives, it comes at the expense of reducing the available data for validation.

From Figure B.5, we observe that, while the coverage of our method remains the same across different N_v, it reduces for other methods. The cause of this phenomenon can be attributed to the fact that we are borrowing the data from the training budget, as it limits the performance of the base model, which in turn limits the auto-labeling objective.

As we increase the percentage of data that can be used to calibrate the model, i.e., ν, we note from Figure B.6 that other methods improve the coverage, which can be understood from the fact that when more data is available for calibrating the model, the model becomes better in terms of the auto-labeling objective. But it's interesting to note that even with a low calibration fraction, our method achieves superior coverage compared to other methods. It is also important to note that the auto-labeling error increases as we increase ν. This is because when ν increases, the number of data points used to estimate the threshold decreases, leading to a less granular and precise threshold.

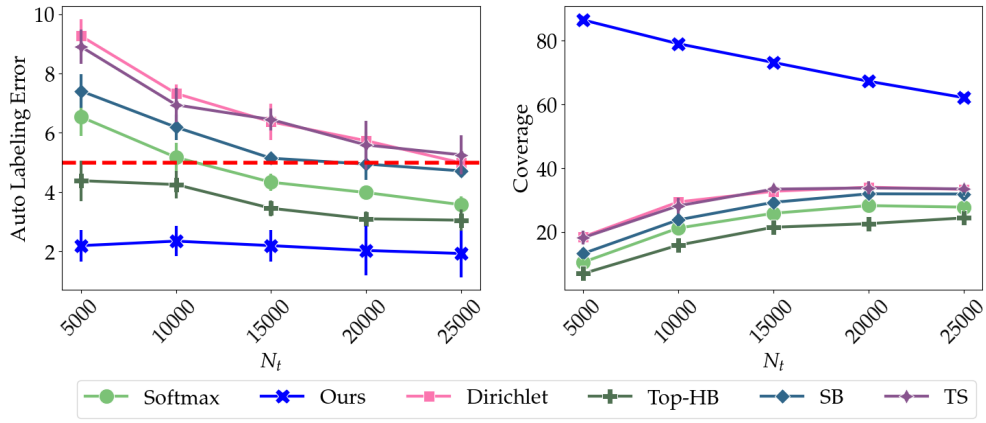


Figure B.4: Autolabeling error and coverage of different post-hoc methods on CIFAR-10 for various N_t

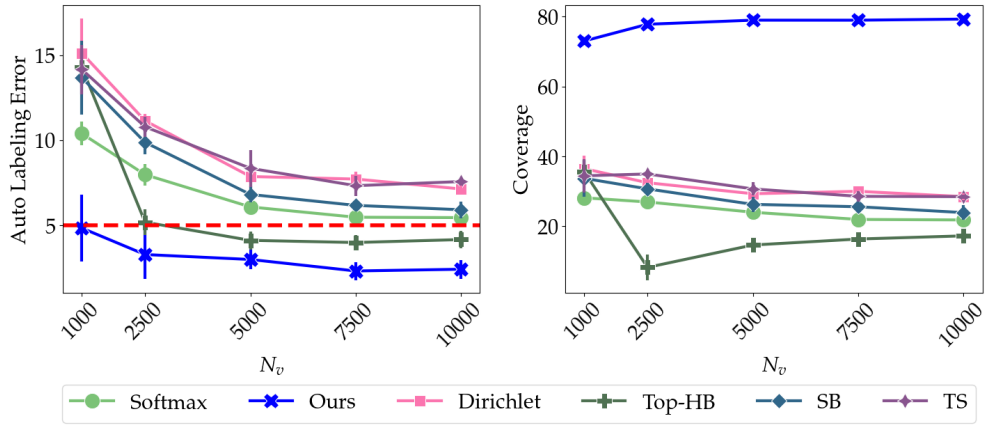


Figure B.5: Autolabeling error and coverage of different post-hoc methods on CIFAR-10 for various N_v

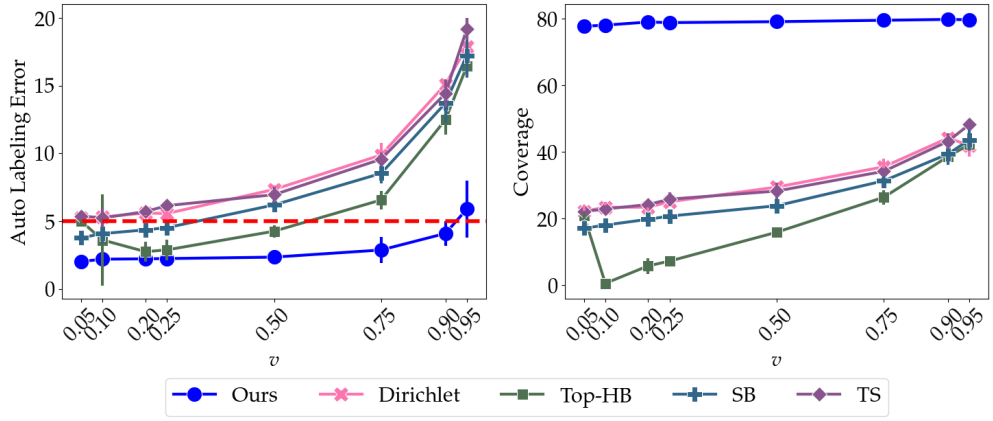


Figure B.6: Autolabeling error and coverage of different post-hoc methods on CIFAR-10 for various ν

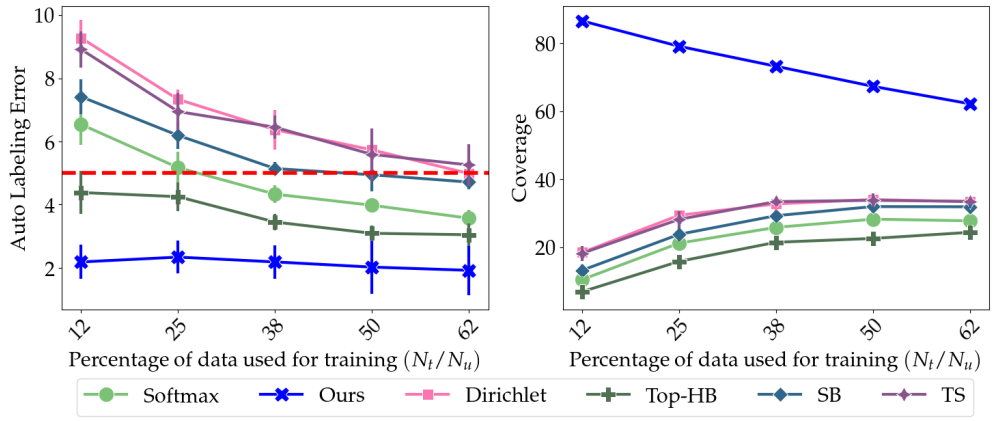


Figure B.7: Auto-labeling error and coverage for different post-hoc methods on CIFAR-10 while we vary N_t. $N_u = 40,000$ is the size of the given unlabeled pool.

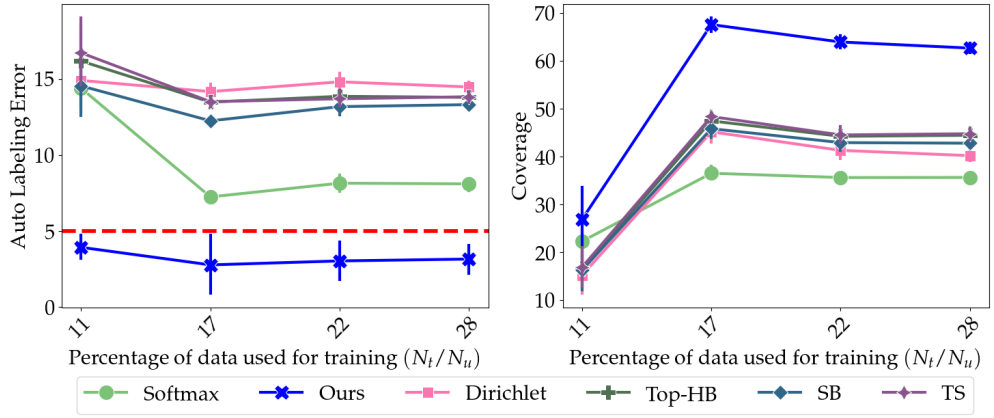


Figure B.8: Auto-labeling error and coverage for different post-hoc methods on Tiny-ImageNet while we vary N_t. $N_u = 90,000$ is the size of the given unlabeled pool.

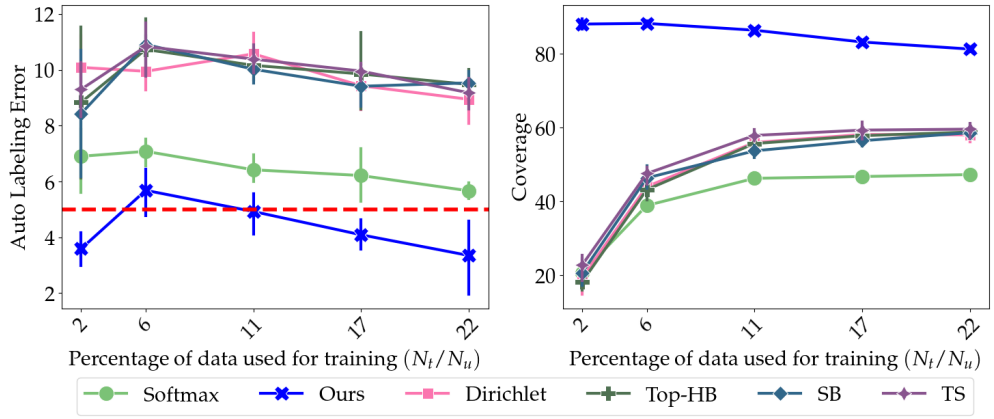


Figure B.9: Auto-labeling error and coverage for different post-hoc methods on 20 Newsgroups while we vary N_t. $N_u = 9,052$ is the size of the given unlabeled pool.

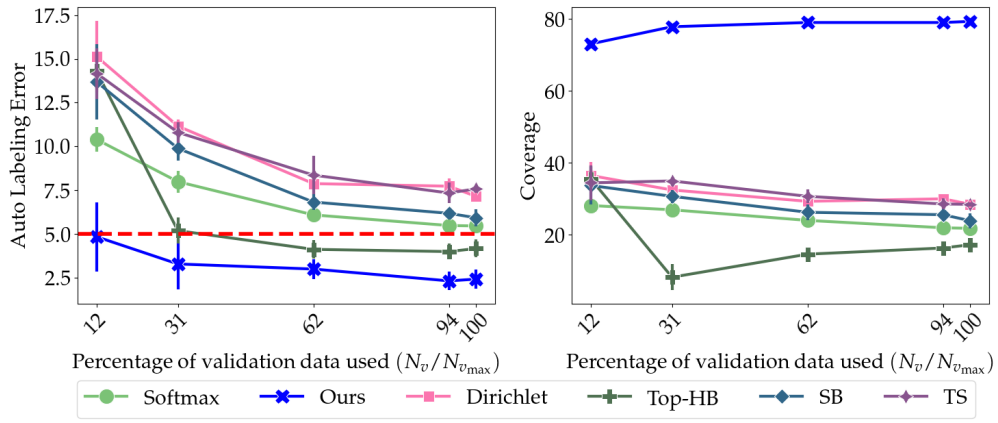


Figure B.10: Auto-labeling error and coverage for different post-hoc methods on CIFAR-10 while we vary N_v. $N_{v_{\max}} = 8,000$ is the maximum number of points available for validation.

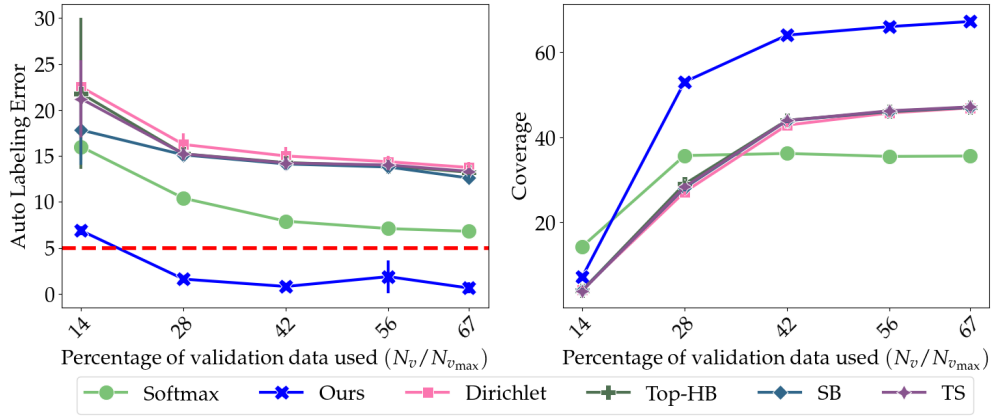


Figure B.11: Auto-labeling error and coverage for different post-hoc methods on Tiny-ImageNet while we vary N_v. $N_{v_{\max}} = 18,000$ is the maximum number of points available for validation.

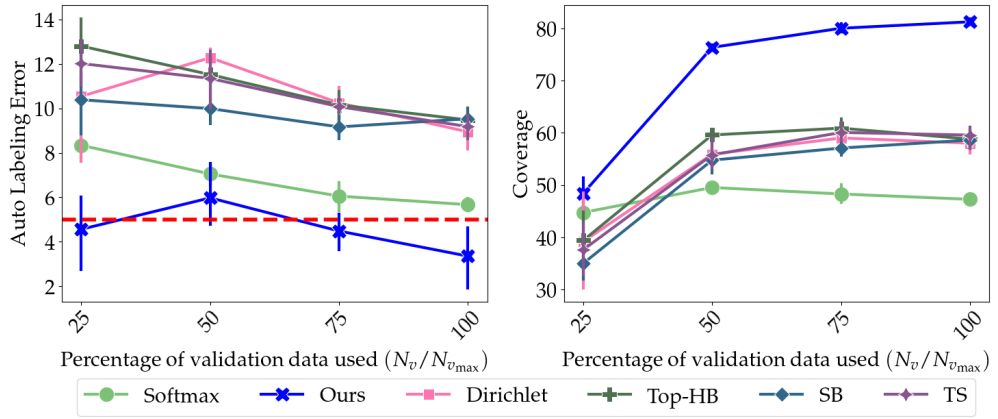


Figure B.12: Auto-labeling error and coverage for different post-hoc methods on 20 Newsgroups while we vary N_v. $N_{v_{\max}} = 1,600$ is the maximum number of points available for validation.

Appendix C

Appendix to Chapter 5

The Appendix is organized as follows. First, we summarize the notations in Table C.1 in Appendix C.1, then we provide formal algorithms in Appendix C.2, and additional experimental results and details are provided in Appendix C.3.

C.1 Glossary

The notations are summarized in Table C.1 below.

C.2 Detailed Algorithms

Algorithm 9: Estimate Pseudolabeling Thresholds Classwise

Input: Confidence function $\widehat{g}_i$, classifier $\widehat{h}_i$, Part of validation data $D_{\text{th}}^{(i)}$ for threshold estimation, pseudolabeling error tolerance ϵ, space of thresholds T, label space $\mathcal{Y}$.

Output: Pseudolabeling thresholds $\widehat{\mathbf{t}}_i$, where $\widehat{\mathbf{t}}_i[y]$ is the threshold for class y.
for $y \in \mathcal{Y}$ **do**
  # Extract the set of points $D_{\text{th}}^{(y)}$ for which the groundtruth class is y.
  $D_{\text{th}}^{(y)} \leftarrow \{(\mathbf{x}', y') \in D_{\text{th}} : y' = y\}$
  $T'_y \leftarrow T \cup \{\infty\}$.
  # Estimate pseudolabeling error at each threshold on class specific data $D_{\text{th}}^{(y)}$. Pick the smallest threshold with the sum of the estimated error and C_1 times the std. deviation is below ϵ. Here C_1 is set to 0.25 and $\widehat{\sigma}(z) = \sqrt{z(1-z)}$.
  $\widehat{\mathbf{t}}_i[y] \leftarrow \min\{t \in T'_y : \widehat{\mathcal{E}}(\widehat{g}_i, t \mid \widehat{h}_i, D_{\text{th}}^{(y)}) + C_1\widehat{\sigma}(\widehat{\mathcal{E}}(\widehat{g}_i, t \mid \widehat{h}_i, D_{\text{th}}^{(y)})) \leq \epsilon\}$,
end for
return $\widehat{\mathbf{t}}_i$

Algorithm 10: Estimate Pseudolabeling Threshold Jointly for All Classes

Input: Confidence function $\widehat{g}_i$, classifier $\widehat{h}_i$, Part of validation data $D_{\text{th}}^{(i)}$ for threshold estimation, pseudolabeling error tolerance ϵ, space of thresholds T, label space $\mathcal{Y}$.

Output: Pseudolabeling thresholds $\widehat{\mathbf{t}}_i$, where $\widehat{\mathbf{t}}_i[y]$ is the threshold for class y.
$T' \leftarrow T \cup \{\infty\}$
Estimate pseudolabeling error at each threshold on the entire set D_{th}. Pick the smallest threshold with the sum of the estimated error and C_1 times $\widehat{\sigma}$ is below ϵ. Here C_1 is set to 0.25 and $\widehat{\sigma}(z) = \sqrt{z(1-z)}$.
$t \leftarrow \min\{t \in T' : \widehat{\mathcal{E}}(\widehat{g}_i, t \mid \widehat{h}_i, D_{\text{th}}) + C_1\widehat{\sigma}(\widehat{\mathcal{E}}(\widehat{g}_i, t \mid \widehat{h}_i, D_{\text{th}})) \leq \epsilon\}$.
for $y \in \mathcal{Y}$ **do**
  $\widehat{\mathbf{t}}_i[y] \leftarrow t$
end for
return $\widehat{\mathbf{t}}_i$

C.3 Additional Experiments and Details

Compute. We ran all of our experiments on a high-throughput system with various GPUs. Therefore, each individual experiment task may be scheduled among NVIDIA A100 SXM4-40GB, NVIDIA A100 SXM4-80GB, NVIDIA L40, and NVIDIA H100 80GB HBM3. We measured the runtime of our algorithm on a desktop with a single NVIDIA RTX 4090. On CIFAR-10, it took about 0.203 seconds for each iteration for our method and around 0.140 seconds for the baselines. On CIFAR-100, it took about 0.396 seconds for each iteration for our method and around 0.143

Algorithm 11: Pseudolabeling Based SSL with `PabLO`

Input: Labeled data for training D_l, validation data D_{val}, unlabeled pool X_u, error tolerance ϵ, use-accumulation flag, num_iters, batch size B, replication factor μ, weak ω and strong Ω augmentations, number of calibration points N_{cal}, num. of threshold estimation points N_{th}, frequency of invoking `PabLO` F, space of thresholds T, label space $\mathcal{Y}$.

Output: $\widehat{h}_{\text{ssl}}$, model with the best validation accuracy.

$\widetilde{Y}_u^{(0)} \leftarrow [0, 0, \ldots, 0]$, $S_u^{(0)} \leftarrow [0, 0, \ldots, 0]$, $i \leftarrow 1$ ▷Set initial pseudolabels and masks to 0.

D_{cal}, $D_{\text{th}} \leftarrow$ `DrawRandomly`$(D_{\text{val}}, N_{\text{cal}}, N_{\text{th}})$.

while $i \leq$ num_iters **do**

Get batches D_l^b, X_u^b of labeled and unlabeled points, I_u^b denotes indices of points in X_u^b.

D_l^b, X_u^b, $I_u^b \leftarrow$ `DrawRandomBatch`$(\mu D_l, \mu X_u, B)$

$X_{u,w}^b$, $X_{u,s}^b \leftarrow \omega(X_u^b)$, $\Omega(X_u^b)$ ▷Create weak and strong augmentations of X_u^b.

/** Begin Pseudolabeling Block. Perform pseudolabeling using `PabLO`. **/

if $i\%F = 0$ **then**

$\widehat{g}_i, \widehat{\mathbf{t}}_i' \leftarrow$ `SolveOptProblemP1`$(\widehat{h}_i, D_{\text{cal}})$. ▷Get $\widehat{g}_i$ by solving optimization (P5.1).

Estimate pseudolabeling thresholds.

if estimate threshold classwise **then**

$\widehat{\mathbf{t}}_i \leftarrow$ `ClasswiseThreshold`$(\widehat{g}_i, \widehat{h}_i, D_{\text{th}}, \epsilon, T, \mathcal{Y})$ ▷Use Algorithm 9.

else

$\widehat{\mathbf{t}}_i \leftarrow$ `JointThreshold`$(\widehat{g}_i, \widehat{h}_i, D_{\text{th}}, \epsilon, T, \mathcal{Y})$ ▷Use Algorithm 10.

end if

Compute fresh psuedolabels $\widetilde{Y}_u^{(i)}$ and pseudolabeling masks $S_u^{(i)}$ for all points in X_u.

$\widetilde{Y}_u^{(i)} \leftarrow \widehat{h}_i(\omega(X_u))$, $S_u^{(i)} \leftarrow \mathbb{1}(\widehat{g}_i(\omega(X_u)) \geq \widehat{\mathbf{t}})$

if use-accumulation **then**

Apply pseudolabel accumulation if enabled.

$\widetilde{Y}_u^{(i)} \leftarrow S_u^{(i)} \widetilde{Y}_u^{(i)} + (1 - S_u^{(i)}) \widetilde{Y}_u^{(i-1)}$

$S_u^{(i)} \leftarrow S_u^{(i)} \vee S_u^{(i-1)}$

end if

else

$\widehat{g}_i, \widehat{\mathbf{t}}_i = \widehat{g}_{i-1}, \widehat{\mathbf{t}}_{i-1}$

end if

/** End Pseudolabeling Block **/

$\widetilde{Y}_u^b$, $S_u^b \leftarrow \widetilde{Y}_u[I_u^b]$, $S_u[I_u^b]$ ▷Extract pseudolabels and masks for the unlabeled batch.

$\widehat{\mathcal{L}}_s(\widehat{h}_i) \leftarrow$ `supervised_loss`(h, D_l^b)

$\widehat{\mathcal{L}}_u(\widehat{h}_i) \leftarrow$ `unsupervised_loss`$(h, X_{u,w}^b X_{u,s}^b, \widetilde{Y}_u^b, S_u^b)$

$\widehat{\mathcal{L}}(\widehat{h}_i) \leftarrow \widehat{\mathcal{L}}_s(\widehat{h}_i) + \lambda_u \widehat{\mathcal{L}}_u(\widehat{h}_i)$

$\widehat{h}_{i+1} \leftarrow$ `SGD_update`$(\widehat{\mathcal{L}}(\widehat{h}_i))$; $i \leftarrow i + 1$ ▷Perform a gradient descent to get model $\widehat{h}_{i+1}$.

if $i\%$eval_freq $= 0$ **then**

eval_acc $\leftarrow$ `evaluate_model`$(\widehat{h}_i, D_{\text{val}})$

If eval_acc is best so far then $\widehat{h}_{\text{ssl}} = \widehat{h}_i$

end if

end while

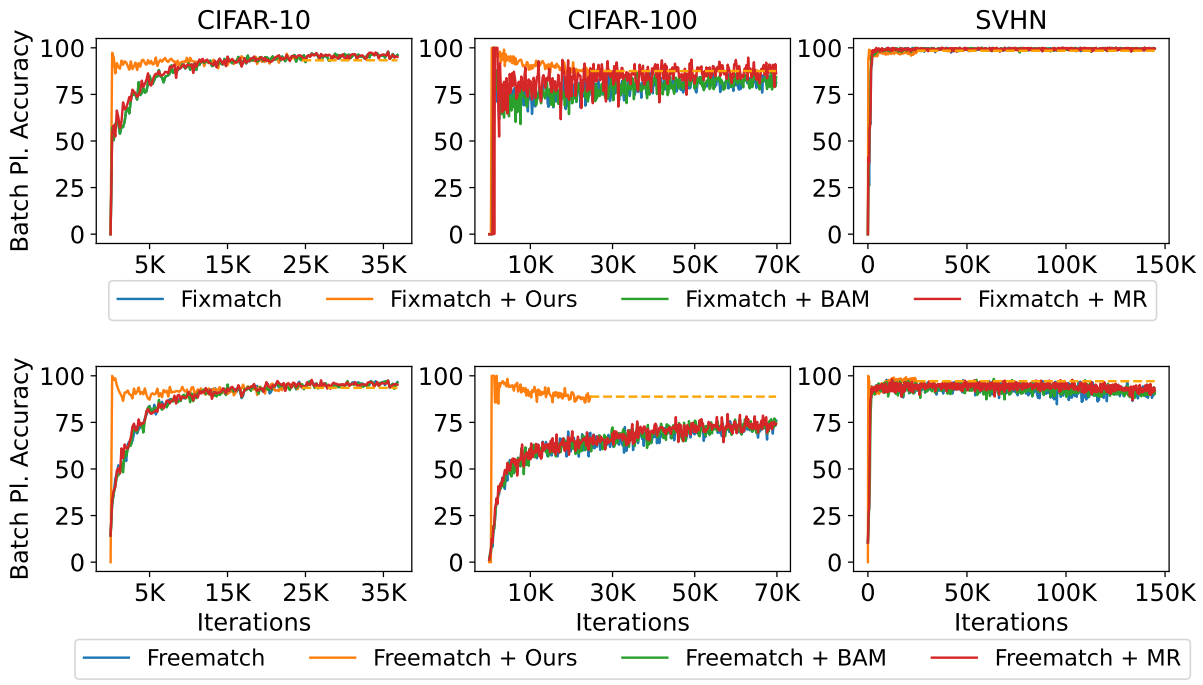


Figure C.1: Batch pseudolabel accuracy of our method and baselines on CIFAR-10, CIFAR-100, and SVHN. We plot the values for every 200 steps.

seconds for the baselines. On SVHN, it took about 1.275 seconds for each iteration for our method and around 0.225 seconds for the baselines.

Hyperparameters. For the baselines, we have used their default settings. To maintain consistency and experiment with the efficiency of the method, we used WRN-28-2, which is 1.4M parameter model, for all the datasets. We summarize the main hyperparameters we have used in our method in Table C.5.

Note that the number of epochs we used to train the function g and to estimate $\mathbf{t}$ is dynamic. That is, its actual value depends on and is proportional to the current number of iterations of the SSL training. More concretely, at iteration i of SSL training, we use $\min(\lfloor i/25 \rfloor, \text{max epoch})$ number of epochs to find g and $\mathbf{t}$.

We additionally conduct the following ablation study to study our technique's dependence on the amount of data used in learning g and thresholds.

A2. How much data is needed to learn the g and $\mathbf{t}$? We take N_{cal} and N_{th} from the

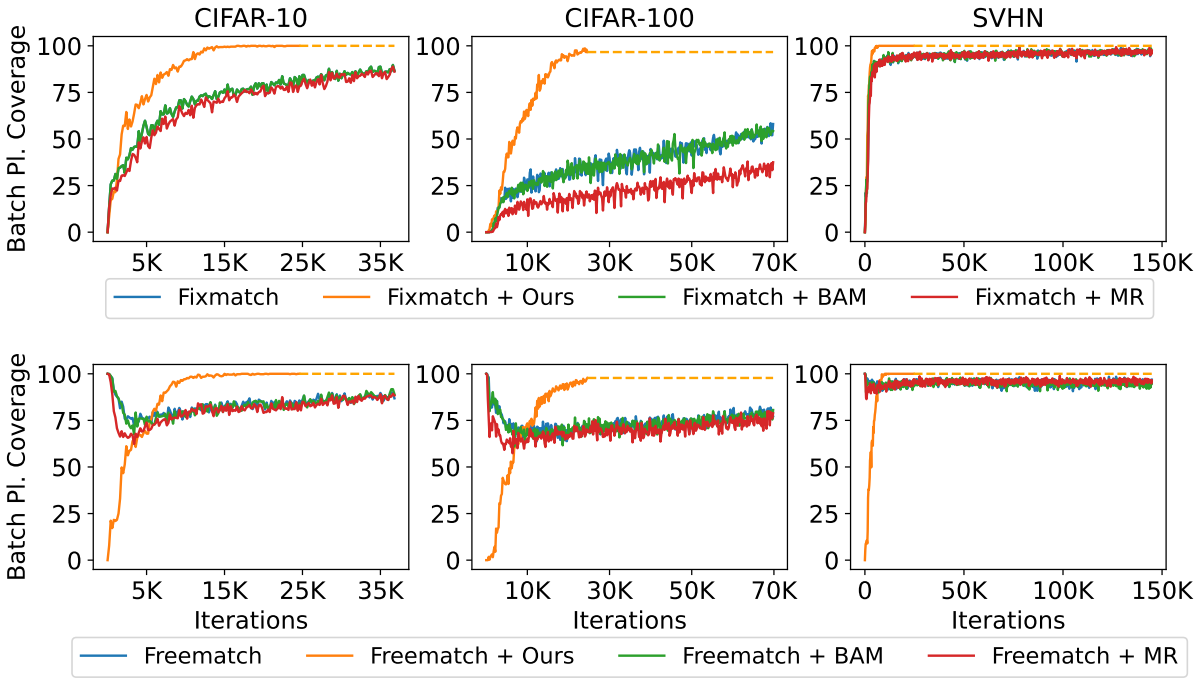


Figure C.2: Batch pseudolabel coverage of our method and baselines on CIFAR-10, CIFAR-100, and SVHN. We plot the values for every 200 steps.

validation data to learn the confidence function g and estimate the thresholds $\mathbf{t}$ respectively. Intuitively, larger values of these should lead to good g and $\mathbf{t}$ that can extract the expected level of pseudolabeling coverage and accuracy from the classifier at hand. However, the task of learning good g and estimating thresholds is not super hard, and we expect it will take fewer samples to be successful. To understand this better we run our method with N_{cal} and N_{th} in $\{250, 500, 750, 1000\}$ on CIFAR-10 setting for 3 random seeds and report the result in Fig C.5. We observe that our method can achieve the desired performance with just 500 labeled points (i.e, 50 labels per class). This is interesting because we can achieve 90% accuracy by just using 250 points (N_l) for training h and a total of 1K for learning g. Refer to Table C.2 for more details.

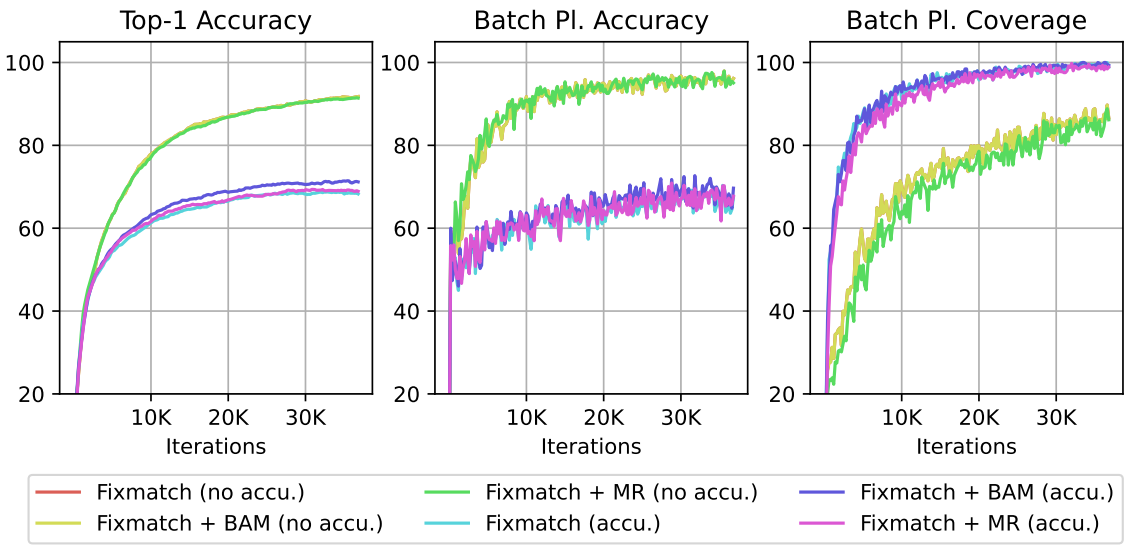


Figure C.3: **(A1.)** Left to Right: Top-1 Accuracy, Batched pseudolabeling Accuracy, and batched pseudolabeling coverage of Fixmatch with and without pseudolabeling accumulation enabled on CIFAR-10. It can be seen that enabling pseudolabeling accumulation worsens the performance of baseline methods in terms of accuracy and coverage.

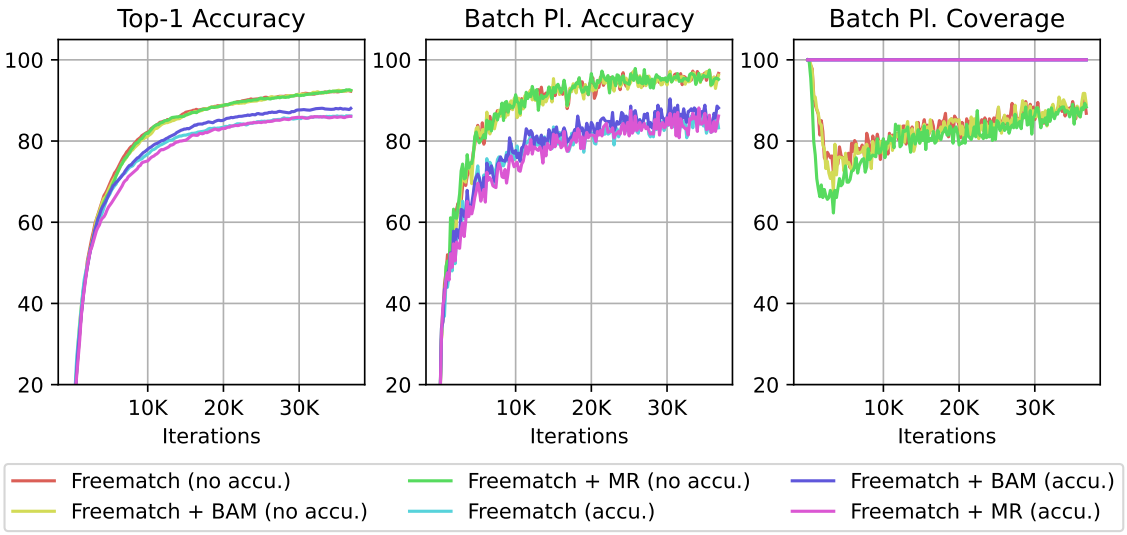


Figure C.4: **(A1.)** Left to Right: Top-1 Accuracy, Batched pseudolabeling Accuracy, and batched pseudolabeling coverage of Freematch with and without pseudolabeling accumulation enabled on CIFAR-10. It can be seen that enabling pseudolabeling accumulation worsens the performance of baseline methods in terms of accuracy and coverage.

Symbol	Definition
$\mathcal{G}$	space of confidence functions.
k	number of classes.
$\mathbf{x}, y$	$\mathbf{x}$ is a datapoint in $\mathcal{X}$ and y is its true label (if available).
$\hbar$	a model $\hbar : \mathcal{X} \to \mathcal{Y}$.
g	confidence function $g : \mathcal{X} \to T^k \subseteq \mathbb{R}^k$
$\widehat{y}$	hard label prediction.
$\widetilde{y}$	$\widehat{y}$ is used as pseudolabel.
$\widehat{h}_{\text{ssl}}$	a best learned model using SSL.
$g_i^\star$	optimal confidence function at i iteration.
$\mathbf{t}_i^*$	optimal threshold at i iteration.
X_u	available unlabeled data drawn from the distribution P_x over $\mathcal{X}$.
X_u^b	batch of unlabeled data.
D_l	set of labeled data points drawn from the distribution P_{xy}.
D_l^b	batch of labeled data.
D_{val}	validation data.
D_{cal}	calibration data; part of val. data used to optimize surrogate functions.
D_{th}	part of validation data to estimate threshold $\mathbf{t}$.
$\mathbf{t}$	k dimensional vector of thresholds representing for k classes.
$\mathbf{t}[y]$	yth entry of $\mathbf{t}$ i.e. the threshold for class y.
n_u	number of unlabeled points, i.e., size of X_u.
N_l	number of labeled points, i.e,. size of D_l.
N_{val}	number of points used for model selection.
N_{test}	number of test data points.
N_{cal}	number of points used for learning the g function.
N_{th}	number of data points used for threshold estimation.
$\widehat{\mathcal{L}}_u$	unsupervised loss with weighted importance λ_u.
$\widehat{\mathcal{L}}_r$	sum of regularization terms with weighted importance λ_r.
$H(y, h, \mathbf{x})$	standard cross-entropy loss.
$S(\mathbf{x}, g, \mathbf{t} \mid h)$	pseudolabeleing mask.
ω	weak transformation, $\omega : \mathcal{X} \mapsto \mathcal{X}$.
Ω	strong transformation, $\Omega : \mathcal{X} \mapsto \mathcal{X}$.
α_o, α_b	average time taken by our method and baseline methods.
$\widehat{\mathcal{P}}(g, \mathbf{t} \mid \hbar, X)$	estimated pseudolabeling coverage, see eq. (5.1).
$\mathcal{P}(g, \mathbf{t} \mid \hbar)$	population level pseudolabeling coverage, see eq. (5.2).
$\widehat{\mathcal{E}}(g, \mathbf{t} \mid \hbar, D)$	estimated pseudolabeling error, see eq. (5.3).
$\mathcal{E}(g, \mathbf{t} \mid \hbar)$	population level pseudolabeling error, see eq. (5.4).
$\widetilde{\mathcal{P}}(g, \mathbf{t} \mid \hbar, D)$	surrogate estimated pseudolabeling coverage, see eq. (5.6).
$\widetilde{\mathcal{E}}(g, \mathbf{t} \mid \hbar, D)$	surrogate estimated pseudolabeling error, see eq. (5.7).
λ	hyperparamter controlling the importance of coverage and error in (P5.1).

Table C.1: Glossary of variables and symbols used in this paper.

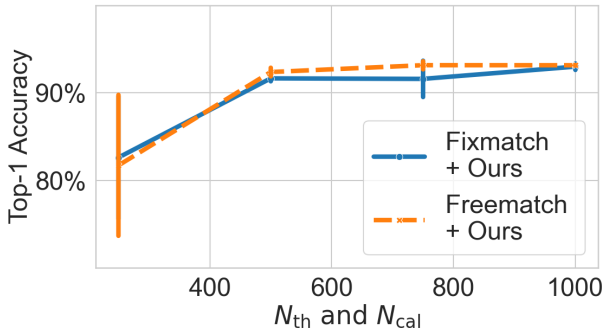


Figure C.5: Top-1 accuracy of our method with different N_{th} and N_{cal}.

Method	$N_{\text{cal}} = N_{\text{th}} = 250$	$N_{\text{cal}} = N_{\text{th}} = 500$	$N_{\text{cal}} = N_{\text{th}} = 750$
Fixmatch + Ours	82.67 ± 7.08	91.74 ± 0.41	91.66 ± 2.11
Freematch + Ours	82.13 ± 7.93	92.33 ± 0.49	93.20 ± 0.53

Table C.2: Results on CIFAR-10 with varying N_{cal} and N_{th}.

Method	$\epsilon = 0.01$	$\epsilon = 0.1$	$\epsilon = 0.2$	$\epsilon = 0.4$
Fixmatch + Ours	93.05 ± 0.54	91.54 ± 0.95	88.35 ± 2.90	56.72 ± 22.25
Freematch + Ours	92.11 ± 1.18	92.31 ± 0.16	83.89 ± 10.36	52.17 ± 25.36

Table C.3: Results on CIFAR-10 with varying ϵ.

Method	$\epsilon = 0.01$	$\epsilon = 0.1$	$\epsilon = 0.2$	$\epsilon = 0.4$
Fixmatch + Ours	69.19 ± 1.13	65.01 ± 0.34	53.88 ± 8.15	23.58 ± 18.21
Freematch + Ours	70.13 ± 0.67	64.95 ± 1.41	59.83 ± 1.32	24.09 ± 17.22

Table C.4: Results on CIFAR-100 with varying ϵ.

Method	**Hyperparameter**	**Values**
Learning g function	optimizer	SGD
	learning rate	0.01
	batch size	64
	max epoch	500
	weight decay	0.01
	momentum	0.9
Estimating $\mathbf{t}$	optimizer	SGD
	learning rate	0.01
	batch size	64
	max epoch	500
	weight decay	0.01
	momentum	0.9

Table C.5: Hyperparameters used for our method.

Appendix D

Appendix to Chapter 6

The Appendix is organized as follows. First, we provide a glossary that summarizes the notation we use throughout the paper. Afterwards, we provide the proofs for the finite-valued metric space cases. We continue with the proofs and additional discussion for the manifold-valued label spaces. Finally, we give some additional explanations for pseudo-Euclidean spaces.

D.1 Glossary

The glossary is given in Table D.1 below.

D.2 Proofs for Parameter Estimation Error in Discrete Spaces

We introduce results leading to the proofs of the theorems for the finite-valued metric space case.

Lemma D.2.1. *(Anandkumar et al., 2014b) Let $\hat{\mathbf{T}}^{+}, \hat{\mathbf{T}}^{-}$ be the third-order observed moments for mutually independent labeling functions triplet, as defined in (6.5) using a sufficiently large number n of i.i.d observations drawn from models in equation (6.4). Suppose there are sufficiently many such triplets to cover all labeling functions. Let $\hat{\boldsymbol{\mu}}^{+}_{a,y}, \hat{\boldsymbol{\mu}}^{-}_{a,y}$ be the estimated parameters returned by the algorithm 3 for all $a \in [m]$. Let $\epsilon(d)$ be defined as above in equation (6.6), then*

the following holds with high probability for all labeling functions,

$$||\boldsymbol{\mu}^{+}_{a,y} - \hat{\boldsymbol{\mu}}^{+}_{a,y}||_2 \leq \mathcal{O}(\epsilon(d^{+})) \quad \textit{and} \quad ||\boldsymbol{\mu}^{-}_{a,y} - \hat{\boldsymbol{\mu}}^{-}_{a,y}||_2 \leq \mathcal{O}(\epsilon(d^{-})) \quad \forall a \in [m]\ \forall y \in \mathcal{Y}_s \tag{D.1}$$

Proof. The result follows by first showing that our setting and assumptions imply that the conditions of Theorems 1 and 5 in Anandkumar et al. (2014b) are satisfied, which allows us to adopt their results. We then translate the result in order to state it in terms of the ℓ_2 distance.

The tensor concentration result in Theorem 1 in Anandkumar et al. (2014b) relies heavily on the noise matrices satisfying the Restricted Isometry Property (RIP). The authors make an explicit assumption that the noise model satisfies this condition. In our setting, we have a specific form of the noise model that allows us to show that this assumption is satisfied. The RIP condition is satisfied for sub-Gaussian noise matrices (Baraniuk et al., 2014). Our noise matrices are supported on a discrete space and have bounded entries, and so are sub-Gaussian.

The other required conditions on the norms of factor matrices and the number of latent factors are implied by Assumption 1. Thus, we can adopt the results on recovery of parameters $\boldsymbol{\mu}_{a,y}$ and the prior weights w_y from Anandkumar et al. (2014b). The result gives us for all $a \in [m], y \in \mathcal{Y}_s$,

$$\text{dist}(\boldsymbol{\mu}^{+}_{a,y}, \hat{\boldsymbol{\mu}}^{+}_{a,y}) \leq \mathcal{O}\big(\epsilon(d^{+})\big), \quad \text{dist}(\boldsymbol{\mu}^{-}_{a,y}, \hat{\boldsymbol{\mu}}^{-}_{a,y}) \leq \mathcal{O}\big(\epsilon(d^{-})\big),$$

and

$$|w_y - \hat{w}_y| \leq \mathcal{O}\Big(\max\big(\epsilon(d^{+}), \epsilon(d^{-})\big)/k\Big),$$

where $\text{dist}(\mathbf{u}, \mathbf{v})$ is defined as follows. For any $\mathbf{u}, \mathbf{v} \in \mathbb{R}^d$,

$$\text{dist}(\mathbf{u}, \mathbf{v}) = \sup_{\mathbf{z} \perp \mathbf{u}} \frac{\langle \mathbf{z}, \mathbf{v} \rangle}{||\mathbf{z}||_2 ||\mathbf{v}||_2} = \sup_{\mathbf{z} \perp \mathbf{v}} \frac{\langle \mathbf{z}, \mathbf{u} \rangle}{||\mathbf{z}||_2 ||\mathbf{u}||_2}.$$

Next, we translate the result to the Euclidean distance. For $\mathbf{u}, \mathbf{v} \in \mathbb{R}^d$ with $||\mathbf{u}||, ||\mathbf{v}|| = 1$, it is

easy to see that

$$\min_{z\in\{-1,+1\}} ||z\mathbf{u}-\mathbf{v}||_2 \leq \sqrt{2}\,\mathrm{dist}(\mathbf{u},\mathbf{v}).$$

This notion of distance is oblivious to sign recovery. However, when sign recovery is possible, then the Euclidean distance can be bounded as follows,

$$||\mathbf{u}-\mathbf{v}||_2 \leq \sqrt{2}\,\mathrm{dist}(\mathbf{u},\mathbf{v}).$$

Next, we make use of Assumption 2 to recover the signs of $\boldsymbol{\mu}^+, \boldsymbol{\mu}^-$. The assumption bounds the angle between true $\boldsymbol{\mu}^+_{a,y}$ and $\mathbf{y}^+$ between $[0, \pi/2 - c)$ for some sufficiently small $c \in (0, \pi/4]$ such that $\sin(c) > \max(\epsilon_0(d^+), \epsilon_0(d^-))$, where $\epsilon_0(d)$ is defined for some $n_0 < n$ samples in equation (6.6). We measure $\omega(\hat{\boldsymbol{\mu}}^+_{a,y}, \mathbf{y}^+)$ and $\omega(-\hat{\boldsymbol{\mu}}^+_{a,y}, \mathbf{y}^+)$ and claim that whichever makes an acute angle with $\mathbf{y}^+$ has the correct sign.

We have that $\omega(\hat{\boldsymbol{\mu}}^+_{a,y}, \mathbf{y}^+) \leq \omega(\hat{\boldsymbol{\mu}}^+_{a,y}, \boldsymbol{\mu}^+_{a,y}) + \omega(\boldsymbol{\mu}^+_{a,y}, \mathbf{y}^+)$. Let $s \in \{-1, +1\}$ be the correct sign, then,

$$\begin{aligned}
\omega(s\hat{\boldsymbol{\mu}}^+_{a,y}, \mathbf{y}^+) &\leq \omega(s\hat{\boldsymbol{\mu}}^+_{a,y}, \boldsymbol{\mu}^+_{a,y}) + \omega(s\boldsymbol{\mu}^+_{a,y}, \mathbf{y}^+) \\
&\leq \sin^{-1}(\epsilon(d^+)) + \pi/2 - c \\
&< \pi/2 - \left(\sin^{-1}\left(\max(\epsilon_0(d^+), \epsilon_0(d^-))\right) - \sin^{-1}\left(\epsilon(d^+)\right)\right) \\
&< \pi/2 \quad \text{since } \sin^{-1} \text{ is an increasing function in the domain under consideration.}
\end{aligned}$$

With the correct sign $\sin^{-1}(\epsilon(d^+)) < \pi/2$ and so is $\omega(s\hat{\boldsymbol{\mu}}^+_{a,y}, \mathbf{y}^+)$. Thus with incorrect sign $\omega(-s\hat{\boldsymbol{\mu}}^+_{a,y}, \mathbf{y}^+) > \pi/2$.

Hence, after disambiguating the signs we have,

$$||\boldsymbol{\mu}^+_{a,y} - \hat{\boldsymbol{\mu}}^+_{a,y}||_2 \leq \mathcal{O}(\mathrm{dist}(\boldsymbol{\mu}^+_{a,y}, \boldsymbol{\mu}^-_{a,y})) \leq \mathcal{O}(\epsilon(d^+))$$

and similarly for $\boldsymbol{\mu}^-_{a,y}$. Next with n, d sufficiently large such that $\epsilon(d^+), \epsilon(d^-) \leq 1$, the result holds for squared distances. □

Theorem 6.3.1. *Let $\hat{\boldsymbol{\mu}}^+_{a,y}, \hat{\boldsymbol{\mu}}^-_{a,y}$ be the estimates of $\boldsymbol{\mu}^+_{a,y}, \boldsymbol{\mu}^-_{a,y}$ returned by Algorithm 3 with input $\hat{\mathbf{T}}^+, \hat{\mathbf{T}}^-$ constructed using isometric pseudo-Euclidean embeddings (in $\mathbb{R}^{d^+,d^-}$). Suppose Assumptions 1 and 2 are met, a sufficiently large number of samples n are drawn from the model in (6.3), and $k = |\mathcal{Y}_s|$. Then there exists a constant $C_0 > 0$ such that with high probability $\forall a \in [m]$ and $y \in \mathcal{Y}_s$,*

$$|\theta_a - \hat{\theta}_a| \leq C_0 \Big| \mathbb{E}[d^2_{\mathcal{Y}}(\lambda_a, y)] - \hat{\mathbb{E}}[d^2_{\mathcal{Y}}(\lambda_a, y)] \Big| \leq \epsilon(d^+) + \epsilon(d^-),$$

where

$$\epsilon(d) := \begin{cases} \tilde{\mathcal{O}}\left(k\sqrt{\frac{d}{n}}\right) + \tilde{\mathcal{O}}\left(\frac{\sqrt{k}}{d}\right) & \text{if } \sigma^2 = \Theta(1), \\ \tilde{\mathcal{O}}\left(\sqrt{\frac{k}{n}}\right) + \tilde{\mathcal{O}}\left(\frac{\sqrt{k}}{d}\right) & \text{if } \sigma^2 = \Theta(\frac{1}{d}). \end{cases} \tag{6.6}$$

Proof. We prove this by using the bounds on errors in the estimates of $\boldsymbol{\mu}^+_{a,y}$ and $\boldsymbol{\mu}^-_{a,y}$ from Lemma D.2.1. We proceed by bounding the errors in two parts for $\mathbb{E}[d^2_\phi(\boldsymbol{\lambda}^+_a, \mathbf{y}^+)]$ and $\mathbb{E}[d^2_\phi(\boldsymbol{\lambda}^-_a, \mathbf{y}^-)]$ separately and then combine them to get the bound on overall parameter estimation error.

We first bound the error for $\mathbb{E}[d^2_\phi(\boldsymbol{\lambda}^+_a, \mathbf{y}^+)]$. The true mean parameter (i.e., the true expected squared distance) can be expanded as follows:

$$\begin{aligned} \mathbb{E}[d^2_\phi(\boldsymbol{\lambda}^+_a, \mathbf{y}^+)] &= \mathbb{E}\Big[||\boldsymbol{\lambda}^+_a||_2^2 + ||\mathbf{y}^+||_2^2 - 2\langle \boldsymbol{\lambda}^+_a, \mathbf{y}^+ \rangle\Big], \\ &= \mathbb{E}_{\boldsymbol{\lambda}}[||\boldsymbol{\lambda}^+_a||_2^2] + \mathbb{E}_{\mathbf{y}}[||\mathbf{y}^+||_2^2] - 2\mathbb{E}_{\mathbf{y}}[\langle \boldsymbol{\mu}^+_{a,y}, \mathbf{y}^+ \rangle]. \end{aligned}$$

The estimate $\hat{\mathbb{E}}_{\boldsymbol{\lambda}}[||\boldsymbol{\lambda}^+_a||_2^2]$ is computed empirically. The first term is estimated observed LF outputs, i.e. $\hat{\mathbb{E}}_{\boldsymbol{\lambda}}[||\boldsymbol{\lambda}^+_a||_2^2] = \frac{1}{n}\sum_{i=1}^n ||\boldsymbol{\lambda}^{(i),+}_a||_2^2$. The second term is computed by using the estimated prior on the labels, and for the last term, we plug in the estimate of $\boldsymbol{\mu}^+_{a,y}$ computed using the tensor-decomposition algorithm. Putting them all together, we have the following estimator:

$$\hat{\mathbb{E}}[d^2_\phi(\boldsymbol{\lambda}^+_a, \mathbf{y}^+)] = \hat{\mathbb{E}}_{\boldsymbol{\lambda}}[||\boldsymbol{\lambda}^+_a||_2^2] + \hat{\mathbb{E}}_{\mathbf{y}}[||\mathbf{y}^+||_2^2] - 2\hat{\mathbb{E}}_{\mathbf{y}}\langle \hat{\boldsymbol{\mu}}^+_{a,y}, \mathbf{y}^+ \rangle.$$

We want to bound the error of our estimator, i.e., the difference $|\mathbb{E}[d^2_\phi(\boldsymbol{\lambda}^+_a, \mathbf{y})] - \hat{\mathbb{E}}[d^2_\phi(\boldsymbol{\lambda}^+_a, \mathbf{y})]|$.

First, consider the following,

$$
\begin{aligned}
|\mathbb{E}_{\mathbf{y}}\langle \boldsymbol{\mu}_{a,y}^{+}, \mathbf{y}^{+}\rangle - \hat{\mathbb{E}}_{\mathbf{y}}\langle \hat{\boldsymbol{\mu}}_{a,y}^{+}, \mathbf{y}^{+}\rangle| &= \sum_{y} \Big|\big\langle \big(w_y \boldsymbol{\mu}_{a,y}^{+} - \hat{w}_y \hat{\boldsymbol{\mu}}_{a,y}^{+}\big), \mathbf{y}^{+}\big\rangle\Big| \\
&= \sum_{y} \Big|\big\langle \big(w_y \boldsymbol{\mu}_{a,y}^{+} - w_y \hat{\boldsymbol{\mu}}_{a,y}^{+} + w_y \hat{\boldsymbol{\mu}}_{a,y}^{+} - \hat{w}_y \hat{\boldsymbol{\mu}}_{a,y}^{+}\big), \mathbf{y}^{+}\big\rangle\Big| \\
&\leq \sum_{y} \big|w_y \big\langle \big(\boldsymbol{\mu}_{a,y}^{+} - \hat{\boldsymbol{\mu}}_{a,y}^{+}\big), \mathbf{y}^{+}\big\rangle\big| + \sum_{y} \mathcal{O}(\epsilon(d^{+})/k)\big|\langle \hat{\boldsymbol{\mu}}_{a,y}^{+}, \mathbf{y}^{+}\rangle\big| \\
&\leq \sum_{y} \big|w_y \big\langle \big(\boldsymbol{\mu}_{a,y}^{+} - \hat{\boldsymbol{\mu}}_{a,y}^{+}\big), \mathbf{y}^{+}\big\rangle\big| + \mathcal{O}(\epsilon(d^{+})) \\
&\leq \sum_{y} w_y ||\boldsymbol{\mu}_{a,y}^{+} - \hat{\boldsymbol{\mu}}_{a,y}^{+}||_2 ||\mathbf{y}^{+}||_2 + \mathcal{O}(\epsilon(d^{+})) \\
&\leq \mathcal{O}(\epsilon(d^{+})).
\end{aligned}
$$

Here we used $||\boldsymbol{\mu}_{a,y}^{+} - \hat{\boldsymbol{\mu}}_{a,y}^{+}||_2 \leq \mathcal{O}(\epsilon(d^{+}))$ and $||\boldsymbol{\mu}_{a,y}^{+}||_2, ||\hat{\boldsymbol{\mu}}_{a,y}^{+}||_2 = 1, ||\mathbf{y}^{+}||_2 \leq 1, ||\boldsymbol{\lambda}_{a}^{+}||_2^2 \leq 1$ and $|w_y - \hat{w}_y| \leq \mathcal{O}(\epsilon(d^{+}))/k$. Similarly,

$$
\big|\hat{\mathbb{E}}_{\mathbf{y}}[||\mathbf{y}^{+}||_2^2] - \mathbb{E}_{\mathbf{y}}[||\mathbf{y}^{+}||_2^2]\big| = \big|\sum_{y} w_y ||\mathbf{y}^{+}||_2^2 - \hat{w}_y ||\mathbf{y}^{+}||_2^2\big| \leq \sum_{y} |w_y - \hat{w}_y| \cdot ||\mathbf{y}^{+}||_2^2 \leq \mathcal{O}(\epsilon(d^{+})) \tag{D.2}
$$

Hence the parameter estimator error,

$$
\begin{aligned}
\Big|\mathbb{E}[d_{\phi}^{2}(\boldsymbol{\lambda}_{a}^{+}, \mathbf{y})] - \hat{\mathbb{E}}[d_{\phi}^{2}(\boldsymbol{\lambda}_{a}^{+}, \mathbf{y})]\Big| &\leq \Big|\mathbb{E}_{\boldsymbol{\lambda}}[||\boldsymbol{\lambda}_{a}^{+}||_2^2 - \hat{\mathbb{E}}_{\boldsymbol{\lambda}}[||\boldsymbol{\lambda}_{a}^{+}||_2^2]\Big| + 2|\mathbb{E}_{\mathbf{y}}\langle \boldsymbol{\mu}_{a,y}^{+}, \mathbf{y}^{+}\rangle - \hat{\mathbb{E}}_{\mathbf{y}}\langle \hat{\boldsymbol{\mu}}_{a,y}^{+}, \mathbf{y}^{+}\rangle| \\
&\qquad + \big|\hat{\mathbb{E}}_{\mathbf{y}}[||\mathbf{y}^{+}||_2^2] - \mathbb{E}_{\mathbf{y}}[||\mathbf{y}^{+}||_2^2]\big| \\
&\leq \mathcal{O}(1/\sqrt{n}) + \mathcal{O}(\epsilon(d^{+})) + \mathcal{O}(\epsilon(d^{+})) \\
&\leq \mathcal{O}(\epsilon(d^{+})).
\end{aligned}
$$

In the second step, we bound the first term by $\mathcal{O}(1/\sqrt{n})$ via standard concentration inequalities.

Doing the same calculations for $\boldsymbol{\lambda}_a^-$, we obtain

$$\left|\mathbb{E}[d_\phi^2(\boldsymbol{\lambda}_a^-, \mathbf{y})] - \hat{\mathbb{E}}[d_\phi^2(\boldsymbol{\lambda}_a^-, \mathbf{y})]\right| \leq \mathcal{O}(\epsilon(d^-)).$$

The overall error in mean parameters is then

$$\begin{aligned}\left|\mathbb{E}[d_\phi^2(\boldsymbol{\lambda}_a, \mathbf{y})] - \hat{\mathbb{E}}[d_\phi^2(\boldsymbol{\lambda}_a, \mathbf{y})]\right| &\leq \left|\mathbb{E}[d_\phi^2(\boldsymbol{\lambda}_a^+, \mathbf{y})] - \hat{\mathbb{E}}[d_\phi^2(\boldsymbol{\lambda}_a^+, \mathbf{y})]\right| + \\ &\quad \left|\mathbb{E}[d_\phi^2(\boldsymbol{\lambda}_a^-, \mathbf{y})] - \hat{\mathbb{E}}[d_\phi^2(\boldsymbol{\lambda}_a^-, \mathbf{y})]\right|, \\ &\leq \mathcal{O}(\epsilon(d^+)) + \mathcal{O}(\epsilon(d^-)).\end{aligned}$$

Next, we use a known relation between the mean and the canonical parameters of the exponential model to get the result in terms of the canonical parameters:

$$|\theta_a - \hat{\theta}_a| \leq \frac{1}{e_{\min}(A_a(\theta))} \left|\mathbb{E}[d_y^2(\lambda_a, y) - \hat{\mathbb{E}}[d_y^2(\lambda_a, y)]\right|.$$

where $A_a(\theta)$ is the log partition function of the label model in (6.3) and $e_{\min}(A_a) = \inf_{\theta \in \Theta} \frac{d^2}{d\theta^2} A_a(\theta)$ over the parameter space Θ. For more details, see Lemma 8 from Fu et al. (2020) and Theorem 4.3 in Shin et al. (2022a). Letting $C_0 = \max_{a \in [m]} e_{\min}(A_a)$ concludes the proof. □

D.3 Proofs for Generalization Error in Discrete Space

In this section, we give the proof for the generalization error bound in the discrete label spaces. We first show that the perturbed (noise-aware) distance function $\tilde{d}_p$ is an unbiased estimator of the true distance. Using this, we show that the noise-aware score function $\tilde{F}_p$ is a good uniform approximation of the score function F. Then we show that the minimizer $\hat{f}_p$ of $\tilde{F}_p$ is close to the minimizer $\hat{f}$ and that this closeness depends on how well $\tilde{F}_p$ approximates F. Next, showing that $\tilde{F}_q$ is a good uniform approximation of $\tilde{F}_p$ using the results from the previous section on parameter recovery leads to the result on the generalization error of $\hat{f}_q$.

Lemma D.3.1. *Let the distribution $\tilde{Y}|Y$ be given by $\mathbf{P}$ a $k \times k$ transition probability matrix*

with $\mathbf{P}_{ij} = \mathbb{P}(\tilde{Y} = y_j | Y = y_i)$ *and suppose* $\mathbf{P}$ *is invertible. Let pseudo-distance* $\tilde{d}_p(T, \tilde{Y} = y_j) := \sum_{i=1}^{k} (\mathbf{P}^{-1})_{ji} d_{\mathcal{Y}}^2(T, Y = y_i) \quad \forall y_j \in \mathcal{Y}_s$*, then,*

$$\mathbb{E}_{\tilde{Y}|Y=y_i}\big[\tilde{d}_p(T, \tilde{Y})\big] = d_{\mathcal{Y}}^2(T, y_i). \tag{D.3}$$

Proof. Here we adopt the same ideas as in Natarajan et al. (2013a) to create the unbiased estimator $\tilde{d}_p$ as follows, First we write the equations for the expectations for each y_i. Which gives us a system of linear equations and solving them for $\tilde{d}_p$ gives us the expression for the unbiased estimator.

$$\mathbb{E}_{\tilde{Y}|Y=y_i}[\tilde{d}_p(T, \tilde{Y})] = P_\theta(\tilde{Y} = y_1 | Y = y_i)\tilde{d}_p(T, \tilde{Y} = y_1) + \ldots + P_\theta(\tilde{Y} = y_j | Y = y_i)\tilde{d}_p(T, \tilde{Y} = y_j) + \ldots$$
$$+ P_\theta(\tilde{Y} = y_k | Y = y_i)\tilde{d}_p(T, \tilde{Y} = y_k) = d_{\mathcal{Y}}^2(T, y_i) \quad \forall i \in [k]$$

Set $\tilde{\mathbf{d}}_p \in \mathbb{R}^k$ with ith entry $\tilde{\mathbf{d}}_p[i]$ given by $\tilde{d}_p(T, \tilde{Y} = y_i)$ and similarly define $\mathbf{d}$ with $\mathbf{d}[i] = d_{\mathcal{Y}}^2(T, y_i)$. Then the above system of linear equations can be expressed as follows,

$$\mathbf{P}\tilde{\mathbf{d}}_p = \mathbf{d} \implies \tilde{\mathbf{d}}_p = \mathbf{P}^{-1}\mathbf{d} \implies \tilde{d}_p(T, \tilde{Y} = y_j) = \sum_{i=1}^{k} (\mathbf{P}^{-1})_{ji} d_{\mathcal{Y}}^2(T, Y = y_i) \forall y_j \in \mathcal{Y}_s$$

□

Next, we show that the noisy score function $\tilde{F}_p$ concentrates around the true score function F for all x and y with high probability.

Lemma D.3.2. *Let* F *and* $\tilde{F}_p$ *be defined as in* (6.9) *and* (6.2) *over* n *i.i.d. samples. Then the following holds for any* $x \in \mathcal{X}, y \in \mathcal{Y}$ *with high probability,*

$$|F(x, y) - \tilde{F}_p(x, y)| \leq \tilde{\mathcal{O}}\Big(\big(1 + \frac{\sqrt{k}}{\sigma_{\min}(\mathbf{P})}\big)\sqrt{\frac{1}{n}}\Big) \quad \forall x \in \mathcal{X}, \forall y \in \mathcal{Y}_s, \tag{D.4}$$

where $\sigma_{\min}(\mathbf{P})$ *is the minimum singular value of* $\mathbf{P}$*.*

Proof. Let $\{y_i\}_{i=1}^n$ be the true labels of points $\{x_i\}_{i=1}^n$ and let the pseudo-label for ith point drawn from the true noise model $\mathbf{P}$ be $\tilde{y}_i$. Let $\tilde{\mathbf{d}}_p \in \mathbb{R}^k$ be a vector such that its i^{th} entry is given as $\tilde{\mathbf{d}}_p[i] = \tilde{d}_p(T, \tilde{Z} = y_i)$, and similarly and $\mathbf{d} \in \mathbb{R}^k$ with $\mathbf{d}[i] = d_Y^2(T, Y = y_i)$. Recall the definitions of the score functions F and $\tilde{F}_p$ for any $x \in \mathcal{X}$ and y in $\mathcal{Y}$,

$$F(x,y) := \frac{1}{n}\sum_{i=1}^{n} \alpha_i(x) d_Y^2(y, y_i), \qquad \tilde{F}_p(x,y) := \frac{1}{n}\sum_{i=1}^{n} \alpha_i(x) \tilde{d}_p(y, \tilde{y}_i).$$

Taking their difference,

$$\begin{aligned}\tilde{F}_p(x,y) - F(x,y) &= \frac{1}{n}\sum_{i=1}^{n} \alpha_i(x)\Big(\tilde{d}_p(y,\tilde{y}_i) - d_Y^2(y,y_i)\Big), \\ &= \frac{1}{n}\sum_{i=1}^{n} \alpha_i(x)\xi(y, y_i, \tilde{y}_i).\end{aligned}$$

Here y, y_i are fixed and the randomness is over $\tilde{y}_i$, thus we can think of $\tilde{y}_i$ as random variable $\tilde{Y}_i$ and take the expectation of ξ over the distribution $\mathbf{P}$. From Lemma D.3.1 we have $\mathbb{E}_{\tilde{Y}|Y=y_i}[\xi(y, y_i, \tilde{Y})] = 0$ and this implies $\mathbb{E}[\tilde{F}_p(x,y) - F(x,y)] = 0$.

Moreover, $\alpha_i(x) \cdot \xi(y, y_i, \tilde{Y}_i)$ are independent random variables and $\alpha_i(x) \leq 1$. The ξ are bounded as follows as long as the spectral decomposition of $\mathbf{P}$ is not arbitrary,

$$\max_{z \in \mathcal{Y}_s} \tilde{d}_p(y,z) = ||\tilde{\mathbf{d}}_p||_\infty = ||\mathbf{P}^{-1}\mathbf{d}||_\infty \leq ||\mathbf{P}^{-1}||_\infty ||\mathbf{d}||_\infty.$$

Now using the fact that $||\mathbf{d}||_\infty \leq 1$ and properties of matrix norms we get,

$$||\mathbf{P}^{-1}||_\infty ||\mathbf{d}||_\infty \leq ||\mathbf{P}^{-1}||_\infty \leq \sqrt{k}||\mathbf{P}^{-1}||_2 \leq \frac{\sqrt{k}}{\sigma_{\min}(\mathbf{P})}.$$

Moreover, $\forall y, z \in \mathcal{Y}_s, d_Y^2(y,z) \leq 1$ which gives us the magnitude of random variables $\xi(y, z, \tilde{z})$ is upper bounded by $c_1 := 1 + \frac{\sqrt{k}}{\sigma_{\min}(\mathbf{P})}$ $\forall y, z, \tilde{z} \in \mathcal{Y}_s$. Thus, using Hoeffding's inequality and union bound over all $y \in \mathcal{Y}_s$ we get,

$$|\tilde{F}_p(x,y) - F(x,y)| \leq \tilde{\mathcal{O}}\Big(c_1\sqrt{\frac{1}{n}}\Big) \quad \forall y \in \mathcal{Y}_s, x \in \mathcal{X}.$$

Note that the statement holds for $x \in \mathcal{X}$ without requiring an explicit union bound over x. It is because the above concentration depends only on the labels and the events that the above inequality does not hold for any distinct $x_1, x_2 \in \mathcal{X}$ are the same. □

Now, we show that the distance between the minimizer of $\tilde{F}_p$ and F is bounded.

Lemma D.3.3. *Let $\hat{f}$ be the minimizer as defined in (6.2) over the clean labels and let $\hat{f}_p$ (defined in eq. (6.9)) be the minimizer over the noisy labels obtained from conditional distribution $\tilde{Y}|Y$ i.e.* **P** *such that lemma D.3.1, D.3.2 hold, and let the risk function be defined as in (6.1), then with high probability,*

$$d_{\mathcal{Y}}^2\big(\hat{f}_p(x), \hat{f}(x)\big) \leq \tilde{\mathcal{O}}\Big(\frac{c_1}{\beta}\sqrt{\frac{1}{n}}\Big) \quad \forall x \in \mathcal{X}. \tag{D.5}$$

Proof. Recall the definitions,

$$\hat{f}(x) = \arg\min_{y \in \mathcal{Y}} F(x, y) \qquad \hat{f}_p(x) = \arg\min_{y \in \mathcal{Y}} \tilde{F}_p(x, y)$$

Let $d_{\mathcal{Y}}^2(f_1, f_2) = \sup_{x \in \mathcal{X}} d_{\mathcal{Y}}^2\big(f_1(x), f_2(x)\big)$ and let $\mathcal{B}(\hat{f}, r) = \{f : d_{\mathcal{Y}}^2(\hat{f}, f) \leq r\}$ denote the ball of radius r around $\hat{f}$.

From Lemma D.3.2 we know for $t = \tilde{\mathcal{O}}\Big(c_1\sqrt{\frac{1}{n}}\Big)$,

$$F\big(x, f(x)\big) - t \leq \tilde{F}_p\big(x, f(x)\big) \leq F\big(x, f(x)\big) + t \quad \forall f : \mathcal{X} \mapsto \mathcal{Y}_s.$$

From Assumption 6 we have,

$$F\big(x, f(x)\big) \geq F\big(x, \hat{f}(x)\big) + \beta \cdot d_{\mathcal{Y}}^2(f(x), \hat{f}(x)).$$

Combining the two we get a lower bound on $\tilde{F}_p$,

$$\tilde{F}_p(x, f(x)) \geq F\big(x, \hat{f}(x)\big) + \beta \cdot d_{\mathcal{Y}}^2(f(x), \hat{f}(x)) - t.$$

We want to find a sufficiently large ball around $\hat{f}$ such that the minimizer of $\tilde{F}_p$ does not lie outside this ball. To see this let LB and UB denote the above mentioned lower and upper bounds on $\tilde{F}_p$,

$$LB(\tilde{F}_p, f, x) := F\big(x, \hat{f}(x)\big) + \beta \cdot d_Y^2(f(x), \hat{f}(x)) - t.$$
$$UB(\tilde{F}_p, f, x) := F\big(x, f(x)\big) + t.$$

For $f \in \mathcal{B}(\hat{f}, \frac{2t}{\beta})$ and some f' such that

$$\begin{aligned}
UB(\tilde{F}_p, f, x) &\leq LB(\tilde{F}_p, f', x) \quad \forall x, \\
F\big(x, f(x)\big) + t &\leq F\big(x, \hat{f}(x)\big) + \beta \cdot d_Y^2(f'(x), \hat{f}(x)) - t, \\
F\big(x, f(x)\big) - F\big(x, \hat{f}(x)\big) + t &\leq \beta \cdot d_Y^2(f'(x), \hat{f}(x)) - t, \\
\beta d_Y^2(f(x), \hat{f}(x)) + t &\leq \beta \cdot d_Y^2(f'(x), \hat{f}(x)) - t, \\
d_Y^2(f'(x), \hat{f}(x)) &\geq 2t/\beta + d_Y^2(f(x), \hat{f}(x)).
\end{aligned}$$

Thus considering the greatest lower bound, any f' with $d_Y^2(f'(x), \hat{f}(x)) \geq \frac{4t}{\beta}$ cannot be the minimizer of $\tilde{F}_p$, since there exists some other f with smaller distance from $\hat{f}$ that has smaller value compared to f'. □

Next, we show that a good estimate of the true noise matrix $\mathbf{P}$ by $\mathbf{Q}$ leads to $\tilde{F}_q$ being uniformly close to $\tilde{F}_p$.

Lemma D.3.4. *Let* $\mathbf{Q}$, $\mathbf{P}$ *be the distributions defined in equation* (6.7)*, and* $\tilde{d}_q(T, \tilde{Y})$ *be the distance function as in* (6.8)*, if* $\max_{ij} |\mathbf{P}_{ij} - \mathbf{Q}_{ij}| = \epsilon$*,*

$$\big|\tilde{d}_q(y, \tilde{z}_i) - \tilde{d}_p(y, \tilde{z}_i)\big| \leq \mathcal{O}\Big(k^2\Big(\sigma_{\max}(\mathbf{P}) + \frac{\kappa(\mathbf{P})}{\sigma_{\min}(\mathbf{P})}\Big) \cdot \epsilon\Big) \qquad \forall y \in \mathcal{Y}_s. \tag{D.6}$$

Proof. Let $\tilde{\mathbf{d}}_q \in \mathbb{R}^k$ be a vector such that its i^{th} entry is given as $\tilde{\mathbf{d}}_q[i] = \tilde{d}_q(T, \tilde{Z} = y_i)$, and

similarly, let $\tilde{\mathbf{d}}_p \in \mathbb{R}^k$ with $\tilde{\mathbf{d}}_p[i] = \tilde{d}_p(T, \tilde{Y} = y_i)$, and $\mathbf{d} \in \mathbb{R}^k$ with $\mathbf{d}[i] = d^2_y(T, Y = y_i)$. It is easy to see that $\tilde{\mathbf{d}}_q = \mathbf{Q}^{-1}\mathbf{d}$ and $\tilde{\mathbf{d}}_p = \mathbf{P}^{-1}\mathbf{d}$. Now consider the following expectation w.r.t $\mathbf{P}$,

$$\tilde{\mathbf{d}}_q - \tilde{\mathbf{d}}_p = \mathbf{Q}^{-1}\mathbf{d} - \mathbf{P}^{-1}\mathbf{d} = \big(\mathbf{Q}^{-1} - \mathbf{P}^{-1}\big)\mathbf{d}.$$

Let $\Delta\mathbf{P} = \mathbf{P} - \mathbf{Q}$, and using standard matrix inversion results for small perturbations (Demmel, 1992), and $||\mathbf{d}||_\infty \leq 1$ we get the following. As $\max_{ij}(\Delta\mathbf{P})_{ij} \leq \epsilon$, we have $||\Delta\mathbf{P}||_2 \leq ||\Delta\mathbf{P}||_F \leq \epsilon k$

$$\begin{aligned}
||\tilde{\mathbf{d}}_p - \tilde{\mathbf{d}}_q||_\infty &\leq ||(\mathbf{P} + \Delta\mathbf{P})^{-1} - \mathbf{P}^{-1}||_\infty ||\mathbf{d}||_\infty, \\
&\leq \sqrt{k}||(\mathbf{P} + \Delta\mathbf{P})^{-1} - \mathbf{P}^{-1}||_2 ||\mathbf{d}||_\infty, \\
&= \sqrt{k}\Big(\kappa(\mathbf{P})||\mathbf{P}^{-1}||_2||\Delta\mathbf{P}||_2\Big) + \sqrt{k}\mathcal{O}(||\Delta\mathbf{P}||_2^2), \\
&\leq \sqrt{k} \cdot \kappa(\mathbf{P})||\mathbf{P}^{-1}||_2 \cdot \epsilon k + \mathcal{O}(\epsilon^2 k^{5/2}), \\
&\leq \mathcal{O}\Big(k^{5/2}\Big(1 + \frac{\kappa(\mathbf{P})}{\sigma_{\min}(\mathbf{P})}\Big) \cdot \epsilon\Big) =: c_2.
\end{aligned}$$

□

Lemma D.3.5. *For $\tilde{F}_p$ and $\tilde{F}_q$ defined in (6.9) w.r.t. noise distributions $\mathbf{P}$ and $\mathbf{Q}$ respectively, and let $\max_{ij} |\mathbf{P}_{ij} - \mathbf{Q}_{ij}| \leq \epsilon$ then we have w.h.p.*

$$|\tilde{F}_p(x, y) - \tilde{F}_q(x, y)| \leq \tilde{\mathcal{O}}\Big((2c_1 + c_2)\sqrt{\frac{1}{n}}\Big) \qquad \forall y \in \mathcal{Y}_s, \forall x \in \mathcal{X}. \tag{D.7}$$

with $c_2 = k^{5/2} \cdot \epsilon \cdot \Big(1 + \frac{\kappa(\mathbf{P})}{\sigma_{\min}(\mathbf{P})}\Big)$ and $c_1 = 1 + \frac{\sqrt{k}}{\sigma_{\min}(\mathbf{P})}$,

Proof. Recall, random variables $\tilde{Y}$,$\tilde{Z}$ denote the noisy labels drawn from true and estimated noise distributions $\mathbf{P}, \mathbf{Q}$ respectively and $\tilde{y}_i, \tilde{z}_i$ denote their draw for data point x_i. Note that we do not know $\mathbf{P}$ and $\tilde{y}_i$ in practice and we only know $\mathbf{Q}, \tilde{z}_i$. Here we are using $\mathbf{P}$ and $\tilde{y}_i$ to compare our actual estimates using samples $\tilde{z}_i$ against the estimates one could have obtained from $\tilde{y}_i$.

Recall the definitions,

$$\tilde{F}_p(x,y) := \frac{1}{n}\sum_{i=1}^{n}\alpha_i(x)\tilde{d}_p(y,\tilde{y}_i), \qquad \tilde{F}_q(x,y) := \frac{1}{n}\sum_{i=1}^{n}\alpha_i(x)\tilde{d}_q(y,\tilde{z}_i).$$

Then,

$$\tilde{F}_p(x,y) - \tilde{F}_q(x,y) = \frac{1}{n}\sum_{i=1}^{n}\alpha_i(x)\Big(\tilde{d}_p(y,\tilde{y}_i) - \tilde{d}_q(y,\tilde{z}_i)\Big) = \frac{1}{n}\sum_{i=1}^{n}\alpha_i(x)\xi(y,\tilde{y}_i,\tilde{z}_i).$$

Thus,

$$\begin{aligned}\mathbb{E}_{\tilde{Y},\tilde{Z}|Y=y_i}\big[\tilde{d}_p(y,\tilde{Y}) - \tilde{d}_q(y,\tilde{Z})\big] &= \mathbb{E}_{\tilde{Z}|Y=y_i}[\tilde{d}_q(y,\tilde{Y})] - \mathbb{E}_{\tilde{Z}|Y=y_i}[\tilde{d}_q(y,\tilde{Z})] \\ &= d^2_U(y,y_i) - d^2_U(y,y_i) = 0\end{aligned}$$

Finally $\mathbb{E}_{\tilde{Y},\tilde{Z}}[\xi(y,\tilde{Y},\tilde{Z})] = 0$.

Next,

$$\begin{aligned}\tilde{d}_p(y,\tilde{y}_i) - \tilde{d}_q(y,\tilde{z}_i) &\le |\tilde{d}_p(y,\tilde{y}_i) - \tilde{d}_q(y,\tilde{z}_i)| \\ &\le |\tilde{d}_p(y,\tilde{y}_i) - \tilde{d}_p(y,\tilde{z}_i) + \tilde{d}_p(y,\tilde{z}_i) - \tilde{d}_q(y,\tilde{z}_i)| \\ &\le |\tilde{d}_p(y,\tilde{y}_i) - d^2_U(y,\tilde{z}_i) + d^2_U(y,\tilde{z}_i) - \tilde{d}_p(y,\tilde{z}_i) + \tilde{d}_p(y,\tilde{z}_i) - \tilde{d}_q(y,\tilde{z}_i)| \\ &\le |\tilde{d}_p(y,\tilde{y}_i) - d^2_U(y,\tilde{z}_i)| + |d^2_U(y,\tilde{z}_i) - \tilde{d}_p(y,\tilde{z}_i)| + |\tilde{d}_p(y,\tilde{z}_i) - \tilde{d}_q(y,\tilde{z}_i)| \\ &\le 2c_1 + |\tilde{d}_p(y,\tilde{z}_i) - \tilde{d}_q(y,\tilde{z}_i)| \\ &\le 2c_1 + c_2.\end{aligned}$$

The first two terms are upper bounded as in Lemma D.3.2, and the last term is bounded using Lemma D.3.4. Since $\alpha_i(x) \le 1$ and $|\xi(y,\tilde{y}_i,\tilde{z}_i)|$ are upper bounded by $2c_1 + c_2$ as shown above, we have that $|\alpha_i(x)\cdot\xi(y,\tilde{y}_i,\tilde{z}_i)| \le 2c_1 + c_2$. □

Lemma D.3.6. *Let $\hat{f}_p$ be the minimizer as defined in (6.9) over the noisy labels drawn from* $\mathbf{P}$*, and let $\hat{f}_q$ (defined in eq. (6.9)) be the minimizer over the noisy labels obtained from the*

conditional distribution $\mathbf{Q}$. *Then with high probability,*

$$d_{\mathcal{Y}}^2\big(\hat{f}_q(x), \hat{f}(x)\big) \leq \tilde{\mathcal{O}}\Big(\frac{1}{\beta}(3c_1 + c_2)\sqrt{\frac{1}{n}}\Big) \qquad \forall x \in \mathcal{X}. \tag{D.8}$$

Proof. Let $t_1 = \tilde{\mathcal{O}}\Big(c_1\sqrt{\frac{1}{n}}\Big)$ and $t_2 = \tilde{\mathcal{O}}\Big((2c_1 + c_2)\sqrt{\frac{1}{n}}\Big)$, then combining Lemma D.3.5 and D.3.2 we have,

$$F\big(x, f(x)\big) - t_1 - t_2 \leq \tilde{F}_q\big(x, f(x)\big) \leq F\big(x, f(x)\big) + t_1 + t_2.$$

Then, following the same argument as in Lemma D.3.3, we get the result. □

The following lemmas bound the estimation error between noise matrices $\mathbf{P}$ and $\mathbf{Q}$ using the estimation error in the canonical parameters.

Lemma D.3.7. *The posterior distribution function* $P_{\boldsymbol{\theta}}(Y = y|\Lambda = \Lambda^u)$ *is* $(2, \ell_\infty)-$*Lipshcitz continuous in* $\boldsymbol{\theta}$ *for any* $y \in \mathcal{Y}$ *and* $\Lambda^u \in \mathcal{Y}^m$.

$$|P_{\boldsymbol{\theta}_1}(Y = y|\Lambda = \Lambda^u) - P_{\boldsymbol{\theta}_2}(Y = y|\Lambda = \Lambda^u)| \leq 2||\boldsymbol{\theta}_1 - \boldsymbol{\theta}_2||_\infty \qquad \forall \boldsymbol{\theta}_1, \boldsymbol{\theta}_2 \in \mathbb{R}^m.$$

Proof. Recall the definition of the posterior distribution,

$$P_{\boldsymbol{\theta}}(Y = y_i|\Lambda = \Lambda^u) = \frac{p(Y = y_i)P_{\boldsymbol{\theta}}(\Lambda = \Lambda^u|Y = y_i)}{\sum_{y_j \in \mathcal{Y}} p(Y = y_j)P_{\boldsymbol{\theta}}(\Lambda = \Lambda^u|Y = y_j)}.$$

For convenience let $\mathbf{d}^{(u,i)} \in \mathbb{R}^m$ be such that its a^{th} entry $\mathbf{d}_a^{(u,i)} = d_{\mathcal{Y}}^2(\Lambda_a^u, y_i)$

$$P_{\boldsymbol{\theta}}(Y = y_i|\Lambda = \Lambda^u) = \frac{P(Y = y_i)\exp(-\boldsymbol{\theta}^T\mathbf{d}^{(u,i)})}{\sum_{y_j \in \mathcal{Y}} P(Y = y_j)\exp(-\boldsymbol{\theta}^T\mathbf{d}^{(u,j)})}.$$

Let $Z_2(\boldsymbol{\theta}) = \sum_{y_j \in \mathcal{Y}} P(Y = y_j) \exp(-\boldsymbol{\theta}^T \mathbf{d}^{(u,j)})$, then

$$-\nabla_{\boldsymbol{\theta}} \log(Z_2(\boldsymbol{\theta})) = \frac{\sum_{y_j \in \mathcal{Y}} \mathbf{d}^{(u,j)} P(Y = y_j) \exp(-\boldsymbol{\theta}^T \mathbf{d}^{(u,j)})}{Z_2(\boldsymbol{\theta})} = \mathbb{E}_{Y|\Lambda}[\mathbf{d}].$$

Since distances are upper bounded by 1, $||\mathbf{d}||_\infty \leq 1$, so $||\mathbb{E}_{Y|\Lambda}[\mathbf{d}]||_\infty \leq 1$.

Now,

$$\nabla_{\boldsymbol{\theta}} \log\left(P_{\boldsymbol{\theta}}(Y = y|\Lambda = \Lambda^u)\right) = -\mathbf{d}^{(u,i)} - \nabla_{\boldsymbol{\theta}} \log(Z_2(\boldsymbol{\theta})).$$

Thus $||\nabla_{\boldsymbol{\theta}} \log\left(P_{\boldsymbol{\theta}}(Y = y|\Lambda = \Lambda^u)\right)||_\infty \leq 2$.

$$\implies |\log\left(P_{\boldsymbol{\theta}_1}(Y = y|\Lambda = \Lambda^u)\right) - \log\left(P_{\boldsymbol{\theta}_2}(Y = y|\Lambda = \Lambda^u)\right)| \leq 2||\boldsymbol{\theta}_1 - \boldsymbol{\theta}_2||_\infty.$$

Using the fact that for any $t_1, t_2 \in [0, 1]$ $|t_1 - t_2| \leq |\log(t_1) - \log(t_2)|$, gives us the result.

□

Lemma D.3.8. *The distribution function $P_{\boldsymbol{\theta}}(\Lambda = \Lambda^u|Y = y)$ is $(2, \ell_\infty)-$Lipshcitz continuous in $\boldsymbol{\theta}$ for any $y \in \mathcal{Y}$ and $\Lambda^u \in \mathcal{Y}^m$.*

$$|P_{\boldsymbol{\theta}_1}(\Lambda = \Lambda^u|Y = y) - P_{\boldsymbol{\theta}_2}(\Lambda = \Lambda^u|Y = y)| \leq 2||\boldsymbol{\theta}_1 - \boldsymbol{\theta}_2||_\infty \qquad \forall \boldsymbol{\theta}_1, \boldsymbol{\theta}_2 \in \mathbb{R}^m.$$

Proof. Doing the same steps as in the proof of Lemma D.3.7 gives the result. □

Lemma D.3.9. *For the noise distributions $\mathbf{P}, \mathbf{Q}$ in (6.7) with parameters $\boldsymbol{\theta}$, $\hat{\boldsymbol{\theta}}$ respectively and $\mathcal{Y}$ restricted only to the elements with non-zero prior probability, $\mathcal{Y}' = \{y \in \mathcal{Y} : P(Y = y) > 0\}$ the following holds,*

$$\max_{ij} |\mathbf{P}_{ij} - \mathbf{Q}_{ij}| \leq 4 \cdot k^m ||\boldsymbol{\theta} - \hat{\boldsymbol{\theta}}||_\infty .$$

Proof. It is easy to see that for any two bounded functions f_1, f_2 with $|f_1(x)| \leq 1, |f_2(x)| \leq 1$ and Lipschitz continuous with constants L_1, L_2, the product of them is also Lipschitz continuous but with constant $L_1 + L_2$. Using this fact along with lemma D.3.7 and lemma D.3.8 gives the result,

$$|\mathbf{P}_{ij} - \mathbf{Q}_{ij}| \leq \sum_{\Lambda^u \in \mathcal{Y}'} |P_{\boldsymbol{\theta}}(y_i|\Lambda^u)P_{\boldsymbol{\theta}}(\Lambda^u|y_j) - P_{\hat{\boldsymbol{\theta}}}(y_i|\Lambda^u)P_{\hat{\boldsymbol{\theta}}}(\Lambda^u|y_j)| \leq 4 \cdot k^m ||\boldsymbol{\theta} - \hat{\boldsymbol{\theta}}||_\infty.$$

□

It is important to note that we are restricting the values of y and λ to $\mathcal{Y}'$, which is the set of y with non-zero prior probability, and by our assumption, it is small.

Finally, we restate and prove our generalization error result:

Theorem 6.4.1. *(Generalization Error) Let $\hat{f}$ be the minimizer as defined in (6.2) over the clean labels and let $\hat{f}_q$ (defined in (6.9)) be the minimizer over the noisy labels obtained from inference in Algorithm 3. Suppose Assumptions 4,5,6 hold. Then for $\epsilon_2 = k^{5/2} \cdot \tilde{\mathcal{O}}(\epsilon(d^+) + \epsilon(d^-)) \cdot \left(1 + \frac{\kappa(\mathbf{P})}{\sigma_{\min}(\mathbf{P})}\right)$ and $c_1 = 1 + \frac{\sqrt{k}}{\sigma_{\min}(\mathbf{P})}$, with high probability,*

$$R(\hat{f}_q) \leq R(f^*) + \mathcal{O}(n^{-\frac{1}{4}}) + \tilde{\mathcal{O}}\Big(\frac{c_1}{\beta} n^{-\frac{1}{2}}\Big) + \tilde{\mathcal{O}}\Big(\frac{3\epsilon_2}{\beta} n^{-\frac{1}{2}}\Big). \tag{6.11}$$

Proof. Recall the definition of risk function,

$$R(f) = \mathbb{E}_{x,y}\big[d_{\mathcal{Y}}^2\big(f(x), y\big)\big].$$

$$\begin{aligned}
R(\hat{f}_q) &= \mathbb{E}_{x,y}\big[d_{\mathcal{Y}}^2\big(\hat{f}_q(x), y\big)\big], \\
&\leq \mathbb{E}_{x,y}\big[d_{\mathcal{Y}}^2\big(\hat{f}_q(x), \hat{f}(x)\big) + d_{\mathcal{Y}}^2(\hat{f}(x), y) + 2d_{\mathcal{Y}}(\hat{f}_q(x), \hat{f}(x)) \cdot d_{\mathcal{Y}}(\hat{f}(x), y)\big], \\
&= \mathbb{E}_x[d_{\mathcal{Y}}^2\big(\hat{f}_q(x), \hat{f}(x)\big)] + R(\hat{f}) + \tilde{\mathcal{O}}(n^{-1/4}), \\
&\leq \tilde{\mathcal{O}}\Big(\frac{1}{\beta}(c_1 + c2)\sqrt{\frac{1}{n}} + \frac{c_2}{\beta}\epsilon\Big) + R(\hat{f}) + \tilde{\mathcal{O}}(n^{-1/4}).
\end{aligned}$$

Using the result from Ciliberto et al. (2016),

$$R(\hat{f}) \leq R(f^*) + \mathcal{O}(n^{-1/4}).$$

Combining the two, we get

$$R(\hat{f}_q) \leq R(f^*) + \tilde{\mathcal{O}}(n^{-1/4}) + \tilde{\mathcal{O}}\Big(\frac{1}{\beta}(c_1 + c2)\sqrt{\frac{1}{n}} + \frac{c_3}{\beta}\epsilon)\Big).$$

We get the end result by plugging in the bound on $\epsilon = \max_{ij} ||\mathbf{P} - \mathbf{Q}||$ from Lemma D.3.9 and the bound on parameter recovery error $||\boldsymbol{\theta} - \hat{\boldsymbol{\theta}}||_\infty$ from Theorem 6.3.1.

□

D.4 Proofs for Continuous Label Spaces

Next, we present the proofs for the results in the continuous (manifold-valued) label spaces. We restate the first result on invariance:

Lemma D.4.1. *For $\mathcal{Y} = \mathcal{M}$, a hyperbolic manifold, $y \sim P$ for some distribution P on $\mathcal{M}$ and labeling functions λ_a, λ_b drawn from (6.3), $\mathbb{E}\cosh d_{\mathcal{Y}}(\lambda_a, \lambda_b) = \mathbb{E}\cosh d_{\mathcal{Y}}(\lambda_b, y)\mathbb{E}\cosh d_{\mathcal{Y}}(\lambda_b, y)$, while for $\mathcal{Y} = \mathcal{M}$ a spherical manifold, $\mathbb{E}\cos d_{\mathcal{Y}}(\lambda_a, \lambda_b) = \mathbb{E}\cos d_{\mathcal{Y}}(\lambda_b, y)\mathbb{E}\cos d_{\mathcal{Y}}(\lambda_b, y)$.*

Proof. We start with the hyperbolic law of cosines, which states that

$$\cosh d(\lambda_a, \lambda_b) = \cosh d(\lambda_a, y)\cosh d(\lambda_b, y) + \sinh d(\lambda_a, y)\sinh d(\lambda_b, y)\cos\alpha,$$

where α is the angle between the sides of the triangle formed by $(y, \lambda_)$ and (y, λ_b). We can rewrite this as follows. Let $v_a = \log_y(\lambda_a)$, $v_b = \log_y(\lambda_b)$ be tangent vectors in $T_y M$. Then,

$$\cosh d(\lambda_a, \lambda_b) = \cosh d(\lambda_a, y)\cosh d(\lambda_b, y) + (\sinh \|v_a\| \sinh \|v_b\|)\langle \frac{v_a}{\|v_a\|}, \frac{v_b}{\|v_b\|}\rangle.$$

Next, we take the expectation conditioned on y. The right-most term is then

$$\begin{aligned}
&\mathbb{E}[(\sinh\|v_a\|\sinh\|v_b\|)\langle\frac{v_a}{\|v_a\|},\frac{v_b}{\|v_b\|}\rangle|y] \\
&\quad= \mathbb{E}[(\sinh\|v_a\|\sinh\|v_b\|)|y]\mathbb{E}[\langle\frac{v_a}{\|v_a\|},\frac{v_b}{\|v_b\|}\rangle|y] \\
&\quad= 0,
\end{aligned}$$

where the last equality follows from the fact that v_a and v_b are independent conditioned on y. This leaves us with the cosh product terms. Taking expectation again with respect to y gives the result.

The spherical version of the result is nearly identical, replacing hyperbolic sines and cosines with sines and cosines, respectively. □

Note, in addition, that it is easy to obtain a version of this result for curvatures that are not equal to -1 in the hyperbolic case (or $+1$ in the spherical case).

We will use this result for our consistency result, restated below.

Theorem 6.5.2. *Let $\mathcal{M}$ be a hyperbolic manifold and $\Delta(\delta) = \min_\rho Pr\Big(\forall i, d_Y(\lambda_{a,i},\lambda_{b,i)}) \leq \rho\Big) \geq 1-\delta$ for fixed δ in$(0,1)$ and . Then, there exists a constant C_1 so that with probability at least $1-\delta$, $\mathbb{E}|\hat{\mathbb{E}}d_Y^2(\lambda_a,y)) - \mathbb{E}d_Y^2(\lambda_a,y)| \leq C_1\cosh(\Delta(\delta))^{3/2}/C_0\sqrt{2n}$.*

Proof. First, we will condition on the event that the observed outputs have maximal distance (i.e., diameter) Δ. This implies that our statements hold with high probability. Then, we use McDiarmid's inequality. For each pair of distinct LFs a, b, we have that

$$P\left(\frac{1}{n}|\sum_{i=1}^{n}\cosh(d(\lambda_{a,i},\lambda_{b,i})) - \mathbb{E}\cosh(d(\lambda_a,\lambda_b))| \geq t\right) \leq 2\exp\left(-\frac{2nt^2}{\cosh(\Delta)}\right),$$

Integrating the expression above in t, we obtain

$$\mathbb{E}|\hat{\mathbb{E}}\cosh(d(\lambda_a,\lambda_b)) - \mathbb{E}\cosh(d(\lambda_a,\lambda_b))| \leq \frac{\sqrt{\pi\cosh(\Delta)}}{\sqrt{2n}}. \tag{D.9}$$

Next, we use this to control the gap on our estimator. Recall that using the triplet approach, we estimate

$$\hat{\mathbb{E}}\cosh(d(\lambda_a, y)) = \sqrt{\frac{\hat{\mathbb{E}}\cosh d(\lambda_a, \lambda_b)\hat{\mathbb{E}}\cosh d(\lambda_a, \lambda_c)}{(\hat{\mathbb{E}}d(\lambda_b, \lambda_c))^2}}.$$

For notational convenience, we write $\nu(a)$ for $\mathbb{E}(\cosh(d(\lambda_a, y)))$, $\hat{\nu}(a)$ for its empirical counterpart, and $\nu(a, b)$ and $\hat{\nu}(a, b)$ for the versions between pairs of LFs a, b. Then, the above becomes

$$\hat{\nu}(a) = \sqrt{\frac{\hat{\nu}(a, b)\hat{\nu}(a, c)}{(\hat{\nu}(b, c))^2}}.$$

Note that $\cosh(x) \geq 1$, so that $\hat{\nu}(a, b) \geq 1$ and similarly for the empirical versions. We also have that $\hat{\nu}(a, b) \leq \cosh(\Delta)$. With this, we can begin our perturbation analysis. Applying Lemma 6.5.1, we have that

$$\begin{aligned}
\mathbb{E}|\hat{\nu}(a) - \nu(a)| &= \mathbb{E}\left|\sqrt{\frac{\hat{\nu}(a, b)\hat{\nu}(a, c)}{\hat{\nu}(b, c)^2}} - \sqrt{\frac{\nu(a, b)\nu(a, c)}{\nu(b, c)^2}}\right| \\
&= \mathbb{E}\left|\sqrt{\frac{\hat{\nu}(a, b)\hat{\nu}(a, c)}{\hat{\nu}(b, c)^2}} - \sqrt{\frac{\nu(a, b)\hat{\nu}(a, c)}{\hat{\nu}(b, c)^2}} + \sqrt{\frac{\nu(a, b)\hat{\nu}(a, c)}{\hat{\nu}(b, c)^2}} - \sqrt{\frac{\nu(a, b)\nu(a, c)}{\nu(b, c)^2}}\right| \\
&\leq \mathbb{E}\left|\sqrt{\frac{\hat{\nu}(a, b)\hat{\nu}(a, c)}{\hat{\nu}(b, c)^2}} - \sqrt{\frac{\nu(a, b)\hat{\nu}(a, c)}{\hat{\nu}(b, c)^2}}\right| + \mathbb{E}\left|\sqrt{\frac{\nu(a, b)\hat{\nu}(a, c)}{\hat{\nu}(b, c)^2}} - \sqrt{\frac{\nu(a, b)\nu(a, c)}{\nu(b, c)^2}}\right| \\
&= \mathbb{E}\left|\sqrt{\frac{\hat{\nu}(a, c)}{\hat{\nu}(b, c)^2}}(\sqrt{\hat{\nu}(a, b)} - \sqrt{\nu(a, b)})\right| + \mathbb{E}\left|\sqrt{\frac{\nu(a, b)\hat{\nu}(a, c)}{\hat{\nu}(b, c)^2}} - \sqrt{\frac{\nu(a, b)\nu(a, c)}{\nu(b, c)^2}}\right| \\
&\leq \frac{\sqrt{\pi}\cosh(\Delta^2)}{\sqrt{2n}} + \mathbb{E}\left|\sqrt{\frac{\nu(a, b)\hat{\nu}(a, c)}{\hat{\nu}(b, c)^2}} - \sqrt{\frac{\nu(a, b)\nu(a, c)}{\nu(b, c)^2}}\right|.
\end{aligned}$$

To see why the last step holds, note that $\sqrt{\hat{\nu}(a, c)} \leq \sqrt{\cosh(\Delta)}$, while $\hat{\nu}(b, c) \geq 1$. Next, for $\alpha, \beta \geq 1$, $\sqrt{\alpha} - \sqrt{\beta} = \frac{\alpha - \beta}{\sqrt{\alpha} - \sqrt{\beta}} \leq \alpha - \beta$. This means that $\mathbb{E}|\sqrt{\hat{\nu}(a, b)} - \sqrt{\nu(a, b)}| \leq \mathbb{E}|\hat{\nu}(a, b) - \nu(a, b)| \leq \frac{\sqrt{\pi\cosh(\Delta)}}{\sqrt{2n}}$ using (D.9).

Now we can continue, adding and subtracting as before. We have that

$$\mathbb{E}\left|\sqrt{\frac{\nu(a, b)\hat{\nu}(a, c)}{\hat{\nu}(b, c)^2}} - \sqrt{\frac{\nu(a, b)\nu(a, c)}{\nu(b, c)^2}}\right|$$

$$\leq \mathbb{E}\left|\sqrt{\frac{\nu(a,b)\hat{\nu}(a,c)}{\hat{\nu}(b,c)^2}} - \sqrt{\frac{\nu(a,b)\nu(a,c)}{\hat{\nu}(b,c)^2}}\right| + \mathbb{E}\left|\sqrt{\frac{\nu(a,b)\nu(a,c)}{\hat{\nu}(b,c)^2}} - \sqrt{\frac{\nu(a,b)\nu(a,c)}{\nu(b,c)^2}}\right|$$
$$\leq \frac{\sqrt{\pi}\cosh(\Delta)}{\sqrt{2n}} + \mathbb{E}\left|\sqrt{\frac{\nu(a,b)\nu(a,c)}{\hat{\nu}(b,c)^2}} - \sqrt{\frac{\nu(a,b)\nu(a,c)}{\nu(b,c)^2}}\right|$$
$$\leq \frac{\sqrt{\pi}\cosh(\Delta)}{\sqrt{2n}} + \frac{\sqrt{\pi}\cosh(\Delta)^{3/2}}{\sqrt{2n}}.$$

Putting it all together, with probability at least $1-\delta$,

$$\mathbb{E}|\hat{\mathbb{E}}\cosh(d(\lambda_a, y)) - \mathbb{E}\cosh(d(\lambda_a, y))| \leq \frac{2\sqrt{\pi}\cosh(\Delta) + \sqrt{\pi}\cosh(\Delta)^{3/2}}{\sqrt{2n}}. \tag{D.10}$$

Next, recall that C_0 satisfies $\mathbb{E}|\hat{\mathbb{E}}\cosh(d(\lambda_a,\lambda_b)) - \mathbb{E}\cosh(d(\lambda_a,\lambda_b))| \geq C_0\mathbb{E}|\hat{\mathbb{E}}d(\lambda_a,\lambda_b)) - \mathbb{E}d(\lambda_a,\lambda_b)|$. Thus,

$$\mathbb{E}|\hat{\mathbb{E}}d^2(\lambda_a, y) - \mathbb{E}d^2(\lambda_a, y)| \leq \frac{2\sqrt{\pi}\cosh(\Delta) + \sqrt{\pi}\cosh(\Delta)^{3/2}}{C_0\sqrt{2n}}.$$

This concludes the proof. □

Next, we will prove a simple result that is needed in the proof of Theorem D.4.3. Consider the distribution P of the quantities $\alpha(x)(y)d_Y^2(z,y)$ for some fixed $z \in \mathcal{M}$. We can think of this as the population-level version of sample distances that are observed in the supervised version of the problem. We do not have access to it in our approach; it will be used only as an object in our proof. Recall we set $q = \arg\min_{z\in Y}\mathbb{E}[\alpha(x)(y)d_Y^2(z,y)]$ to be the population-level minimizer. Here we use the notation $\alpha(x)(y)$ to denote the corresponding kernel value at a point y. Finally, let us denote P' to be the distribution over the quantities $\alpha(x)(y)\sum_{a=1}^m \beta_a^2 d_Y^2(z,\lambda_{a,i})$.

Lemma D.4.2. *Let the distributions P and P' be defined as above, with q the minimizer of $\mathbb{E}_P[\alpha(x)(y)d_Y^2(z,y)]$. Suppose that Assumptions 7 and 8 hold. Then, q is also the minimizer of $\mathbb{E}'_P[\alpha(x)(y)\sum_{a=1}^m \beta_a^2 d_Y^2(z,\lambda_{a,i})]$.*

Proof. We will use a simple symmetry argument. First, note that we can write q in the following

way,

$$q = \arg\min_{z \in \mathcal{Y}} \int_{T_q \mathcal{M}} \alpha(x)(\log_q(v)) d^2_{\mathcal{Y}}(z, \exp_q(v)) dP.$$

Since $\mathcal{M}$ is a symmetric manifold, if $v \in T_q\mathcal{M}$, there is an isometry sending v to $-v \in T_q\mathcal{M}$. Using this isometry and Assumption 8, we can also write

$$q = \arg\min_{z \in \mathcal{Y}} \int_{T_q \mathcal{M}} \alpha(x)(\log_q(-v)) d^2_{\mathcal{Y}}(z, \exp_q(-v)) dP.$$

Our approach will be to formulate similar symmetric expressions for the minimizer, but this time for the loss over the distribution P'. We will then be able to show, using triangle inequality, that q remains the minimizer.

We can similarly express the minimizer of the loss for P' as

$$\arg\min_{z \in \mathcal{Y}} \int_{T_q \mathcal{M}} \int_{(T_{\exp_q(v)} \mathcal{M})^{\otimes m})} \alpha(x)(\log_q(v)) \sum_{a=1}^{m} \beta_a^2 d^2_{\mathcal{Y}}(z, \exp_{\exp_q(v)}(v_a)) dP'.$$

Here we have broken down the expectation over P' by applying the tower law; the inner expectation is conditioned on point $\exp_q(v)$ and runs over the labeling function outputs $\lambda_1, \ldots, \lambda_m$.

Again using Assumption 8, we can write the minimizer for the loss over P' as $\arg\min_{z \in \mathcal{Y}} F'(z)$, where

$$F'(z) = \int_{T_q \mathcal{M}} \int_{(T_{\exp_q(-v)} \mathcal{M})^{\otimes m})} \alpha(x)(\log_q(-v)) \sum_{a=1}^{m} \beta_a^2 d^2_{\mathcal{Y}}(z, \exp_{\exp_q(-v)}(-v_a)) dP'.$$

Thus we can also write the minimizer as $\arg\min_{z \in \mathcal{Y}} F'(z)$, where

$$F'(z) = \int_{T_q \mathcal{M}} \int_{(T_{\exp_q(-v)} \mathcal{M})^{\otimes m})} \alpha(x)(\log_q(-v)) \sum_{a=1}^{m} \beta_a^2 d^2_{\mathcal{Y}}(z, \exp_{\exp_q(-v)}(-v_a)) dP'.$$

With this, we can write

$$\begin{aligned}
F'(z) &= \frac{1}{2}\left(\int_{T_q m}\int_{(T_{\exp_q(v)}m)^{\otimes m})}\alpha(x)(\log_q(v))\sum_{a=1}^{m}\beta_a^2 d_Y^2(z,\exp_{\exp_q(v)}(v_a))dP'\right.\\
&\quad\left.+\int_{T_q m}\int_{(T_{\exp_q(-v)}m)^{\otimes m})}\alpha(x)(\log_q(-v))\sum_{a=1}^{m}\beta_a^2 d_Y^2(z,\exp_{\exp_q(-v)}(-v_a))dP'\right)\\
&= \frac{1}{2}\left(\int_{T_q m}\int_{(T_{\exp_q(v)}m)^{\otimes m})}\alpha(x)(\log_q(v))\sum_{a=1}^{m}\beta_a^2 d_Y^2\left((z,\exp_{\exp_q(v)}(v_a))\right.\right.\\
&\quad\left.\left.+\ d_Y^2(z,\exp_{\exp_q(-v)}(PT_{\exp_q(v)\to\exp_q(-v)}(-v_a)))\right)dP'\right),
\end{aligned}$$

where $PT_{p\to s}$ denotes parallel transport from p to s.

Note that q is on the geodesic between $\exp_{\exp_q(v)}(v_a)$ and $\exp_{\exp_q(-v)}(PT_{\exp_q(v)\to\exp_q(-v)}(-v_a))$. We exploit this fact by applying the following squared-distance inequality. For three points p, s, z, from the triangle inequality,

$$d_Y(p,z)+d_Y(s,z)\geq d_Y(p,s).$$

Squaring both sides and applying

$$d_Y^2(p,z)+d_Y^2(s,z)\geq 2d_Y(p,z)d_Y(s,z),$$

we obtain that

$$2(d_Y^2(p,z)+d_Y^2(s,z))\geq d_Y^2(p,s),$$

so that

$$d_Y^2(p,z)+d_Y^2(q,z)\geq\frac{1}{2}d_Y^2(p,q).$$

Setting p to be $\exp_{\exp_q(v)}(v_a)$ and s to be $\exp_{\exp_q(-v)}(PT_{\exp_q(v)\to\exp_q(-v)}(-v_a))$ in the above gives

$$\begin{aligned}
F'(z)\geq\frac{1}{2}\left(\int_{T_q m}\int_{(T_{\exp_q(v)}m)^{\otimes m}}\alpha(x)(\log_q(v))\sum_{a=1}^{m}\beta_a^2\right.\\
\left.\frac{1}{2}d_Y^2(\exp_{\exp_q(v)}(v_a),\exp_{\exp_q(-v)}(PT_{\exp_q(v)\to\exp_q(-v)}(-v_a)))dP'\right).
\end{aligned}$$

Now we can apply the fact that q is on the geodesic to rewrite this as

$$F'(z) \geq \frac{1}{2}\left(\int_{T_q m}\int_{(T_{\exp_q(v)}m)^{\otimes m})} \alpha(x)(\log_q(v))\sum_{a=1}^{m}\beta_a^2\frac{1}{2}4d_Y^2(q,\exp_{\exp_q(v)}(v_a))dP'\right).$$

This is because the length of the geodesic connecting $\exp_{\exp_q(v)}(v_a)$ and $\exp_{\exp_q(-v)}(PT_{\exp_q(v)\to\exp_q(-v)}(-v_a))$ is twice that of the geodesic connecting $\exp_{\exp_q(v)}(v_a)$ to q.

Thus, we have

$$F'(z) \geq F'(q),$$

and we are done. □

Finally, this enables us to prove our main result, Theorem D.4.3, restated below:

Theorem D.4.3. *Let $\mathcal{M}$ be a complete manifold and suppose the assumptions above hold. Then, there exist constants C_3, C_4*

$$\mathbb{E}[d_Y^2(\hat{f}(x),\tilde{f}(x))] \leq \frac{C_3\sigma_o^2}{nk_{\min}} + \frac{C_4\sum_{a=1}^{m}\beta_a^2\hat{\mu}_a^2}{mnk_{\min}}.$$

Proof. We use Lemma D.4.2 and compute a bound on the expected distance from the empirical estimates to the common center. In both cases, the approach is nearly identical to that of Stromme (2020) (proof of Theorem 3.2.1); we include these steps for clarity. Suppose that the minimum and maximum values of α are $\alpha_{\min}$ and $\alpha_{\max}$, respectively.

Then, letting we have that, using the hugging function assumption

$$\|\log_q(\hat{f}(x)) - \log_q(y_i)\|^2 \leq k_{\min}d_Y^2(q,\hat{f}(x)) + d_Y^2(\hat{f}(x),y_i).$$

We also have that

$$\|\log_q(\hat{f}(x)) - \log_q(y_i)\|^2 = d_Y^2(q,\hat{f}(x)) - 2\langle\log_q(\hat{f}(x)),\log_q(y_i)\rangle + d_Y^2(q,y_i).$$

Then,

$$(1 - k_{\min}) d_Y^2(q, \hat{f}(x)) \leq 2\langle \log_q(\hat{f}(x)), \log_q(y_i) \rangle + d_Y^2(\hat{f}(x), y_i) - d_Y^2(q, y_i).$$

Now, multiply each of the equations by α_i and sum over them. In that case, the difference on the right side is non-positive, as $\hat{f}(x)$ is the empirical minimizer. This yields

$$\sum_{i=1}^{n} \alpha(x)_i (1 - k_{\min}) d_Y^2(q, \hat{f}(x)) \leq \sum_{i=1}^{n} \alpha(x)_i 2\langle \log_q(\hat{f}(x)), \log_q(y_i) \rangle.$$

Using the minimum and maximum values of α, and setting $\bar{q} = \sum_{i=1}^{n} \log_q(y_i)$, we get

$$\alpha_{\min}(1 - k_{\min}) d_Y^2(q, \hat{f}(x)) \leq 2\alpha_{\max} \langle \log_q(\hat{f}(x)), \bar{q} \rangle.$$

We can apply Cauchy-Schwarz, simplify, then square, obtaining

$$\alpha_{\min}^2 (1 - k_{\min})^2 d_Y^2(q, \hat{f}(x)) \leq 4\alpha_{\max}^2 \|\bar{q}\|^2.$$

What remains is to take the expectation and use the fact that the tangent vectors summed up to form $\bar{q}$ are independent. This yields

$$\alpha_{\min}^2 (1 - k_{\min})^2 \mathbb{E} d_Y^2(q, \hat{f}(x)) \leq 4\alpha_{\max}^2 \frac{\sigma_o^2}{n}.$$

Thus we obtain

$$\alpha_{\min}^2 (1 - k_{\min})^2 \mathbb{E} d_Y^2(q, \hat{f}(x)) \leq 4\alpha_{\max}^2 \frac{\sigma_o^2}{n},$$

or

$$\mathbb{E} d_Y^2(q, \hat{f}(x)) \leq 4 \frac{\alpha_{\max}^2}{\alpha_{\min}^2} \frac{\sigma_o^2}{n k_{\min}}. \tag{D.11}$$

We use the same approach, but we apply it to the $m \times n$ points given by the LFs drawn from the

distribution P'. This yields

$$\alpha_{\min}^2(1-k_{\min})^2\mathbb{E}d_Y^2(q,\tilde{f}(x)) \leq 4\alpha_{\max}^2\frac{\sum_{i=1}^m \beta_a^2\sigma_a^2}{mn},$$

where σ_a^2 corresponds to the expected squared distance for LF a to q. We bound this with triangle inequality, obtaining $\sigma_a^2 \leq 2\sigma_o^2 + 2\hat{\mu}_a^2$, so that

$$\alpha_{\min}^2(1-k_{\min})^2\mathbb{E}d_Y^2(q,\tilde{f}(x)) \leq 8\alpha_{\max}^2\frac{\sum_{i=1}^m \beta_a^2(\sigma_o+\hat{\mu}_a^2}{mn},$$

or,

$$\mathbb{E}d_Y^2(q,\tilde{f}(x)) \leq 8\frac{\alpha_{\max}^2}{\alpha_{\min}^2}\frac{\sum_{i=1}^m \beta_a^2(\sigma_o+\hat{\mu}_a^2}{mnk_{\min}}. \tag{D.12}$$

Now, again using triangle inequality,

$$\mathbb{E}d_Y^2(\hat{f}(x),\tilde{f}(x)) \leq 2\mathbb{E}d_Y^2(q,\hat{f}(x)) + 2\mathbb{E}d_Y^2(q,\tilde{f}(x)).$$

Plugging (D.12) and (D.11) into this bound produces the result. □

D.5 Additional Details on Continuous Label Space

We provide some additional details on the continuous (manifold-valued) case.

Computing $\Delta(\delta)$ In Theorem 6.5.2, we stated the result in terms of $\Delta(\delta)$, a quantity that trades off the probability of failure δ for the diameter of the largest ball that contains the observed points. Note that if we fix the curvature of the manifold, it is possible to compute an exact bound for this quantity by using formulas for the sizes of balls in d-dimensional manifolds of fixed curvature.

Hugging number Note that it is possible to derive a lower bound on the hugging number as a function of the curvature. The way to do so is to use *comparison theorems* that upper bound triangle edge lengths with those of larger-curvature triangles. This makes it possible to establish a concrete value for $k_{\min}$ as a function of the curvature.

We note, as well, that an upper bound $k_{\max}$ on the hugging number can be obtained by a simple rearrangement of Lemma 6 from Zhang and Sra (2016). This result follows from a curvature lower bound based on the hyperbolic law of cosines; the bound we describe follows from the opposite—an upper bound based on spherical triangles.

β Weights and Suboptimality An intuitive way to think of the estimator we described is the following simple Euclidean version. Suppose we have labeling functions $\lambda_1, \ldots, \lambda_m$ that are equal to $y + \varepsilon_a$, where $\varepsilon_a \sim \mathcal{N}(0, \sigma_a^2)$. In this case, if we seek an unbiased estimator with lowest variance, we require a set of weights β_a so that $\sum_a \beta_a = 1$ and $\mathrm{Var}[\frac{1}{m}\sum_{a=1}^m \beta_a \lambda_a]$ is minimized. It is not hard to derive a closed-form solution for the β_a coefficients as a function of the terms σ_a^2.

Now, suppose we use the same solution, but with noisy estimates $\hat{\sigma}^2$ instead. Our weights $\hat{\beta}$ will yield a suboptimal variance, but this will not affect the scaling of the rate in terms of the number of samples n.

D.6 Extended Background on Pseudo-Euclidean Embeddings

We provide some additional background on pseudo-metric spaces and pseudo-Euclidean embeddings. Our roadmap is as follows. First, we note that pseudo-Euclidean spaces are a particular kind of pseudo-metric space, so we provide additional background and formal definitions for these pseudo-metric spaces. Afterwards, we explain some of the ideas behind pseudo-Euclidean spaces, comparing them to standard Euclidean spaces in the context of embeddings.

D.6.1 Pseudo-metric Spaces

Pseudo-metric spaces generalize metric spaces by removing the requirement that pairs of points at distance zero must be identical:

Definition D.6.1. (**Pseudo-metric Space**) A set $\mathcal{Y}$ along with a distance function $d_{\mathcal{Y}} : \mathcal{Y} \times \mathcal{Y} \mapsto \mathbb{R}^+$ is called pseudo-metric space if $d_{\mathcal{Y}}$ satisfies the following conditions,

$$\forall \mathbf{y}, \mathbf{z} \in \mathcal{Y} \qquad d_{\mathcal{Y}}(\mathbf{y}, \mathbf{z}) = d_{\mathcal{Y}}(\mathbf{y}, \mathbf{z}) \tag{D.13}$$

(Symmetry)

$$\forall \mathbf{y} \in \mathcal{Y} \qquad d_{\mathcal{Y}}(\mathbf{y}, \mathbf{y}) = 0 \tag{D.14}$$

(Reflexivity)

$$\forall \mathbf{x}, \mathbf{y}, \mathbf{z} \in \mathcal{Y} \qquad d_{\mathcal{Y}}(\mathbf{y}, \mathbf{x}) \leq d_{\mathcal{Y}}(\mathbf{y}, \mathbf{z}) + d_{\mathcal{Y}}(\mathbf{x}, \mathbf{z}) \tag{D.15}$$

(Triangle Inequality)

These spaces have additional flexibility compared to standard metric spaces: note that while $d(y, y) = 0$, $d(x, y) = 0$ does not imply that x and y are identical. The downside of using such spaces, however, is that conventional algebra may not produce the usual results. For example, limits where the distance between a sequence of points and a particular point tends to zero do not convey the same information as in standard metric spaces. However, these odd properties do not concern us, as we only use the spaces for representing a set of distances from our given metric space.

A finite pseudo-metric space has $|\mathcal{Y}| < \infty$.

D.6.2 Pseudo-Euclidean Spaces

The following definitions are for *finite-dimensional* vector spaces defined over the field $\mathbb{R}$.

Definition D.6.2. (**Symmetric Bilinear Form / Generalized Inner Product**) For a vector space $\mathcal{Y}$ over the field $\mathbb{R}$, a symmetric bilinear form is a function $\phi : \mathcal{Y} \times \mathcal{Y} \mapsto \mathbb{R}$ satisfying the following properties $\forall y_1, y_2, z, y \in \mathcal{Y}, c \in \mathbb{R}$:

P1) $\phi(y_1 + y_2, y) = \phi(y_1, y) + \phi(y_2, y)$,

P2) $\phi(cy, z) = c\phi(y, z)$,

P3) $\phi(y, z) = \phi(z, y)$.

Definition D.6.3. (**Squared Distance w.r.t.** ϕ) Let V be a real vector space equipped with generalized inner product ϕ, then the squared distance w.r.t. ϕ between any two vectors $\mathbf{y}, \mathbf{z} \in V$ is defined as,

$$||\mathbf{y} - \mathbf{z}||_\phi^2 := \phi(\mathbf{y} - \mathbf{z}, \mathbf{y} - \mathbf{z})$$

This definition also gives a notion of squared length for every $\mathbf{y} \in V$,

$$||\mathbf{y}||_\phi^2 := \phi(\mathbf{y}, \mathbf{y})$$

The inner product can also be expressed in terms of a basis of the vector space V. Let the dimension of $\mathcal{Y}$ be d, and $\{\mathbf{b}_i\}_{i=1}^d$ be a basis of $\mathcal{Y}$, then for any two vectors $\mathbf{y} = [y_1, \dots y_d], \mathbf{z} = [z_1, \dots z_d] \in V$,

$$\phi(\mathbf{y}, \mathbf{z}) = \sum_{i=1}^{d} \sum_{j=1}^{d} y_i z_i \phi(\mathbf{b}_i, \mathbf{b}_j)$$

The matrix $\mathbf{M}(\phi) := [\phi(\mathbf{b}_i, \mathbf{b}_j)]_{1 \le i,j \le d}$ is called *the matrix of* ϕ w.r.t the basis $\{\mathbf{b}_i\}_{i=1}^d$ It gives a convenient way to express the inner product as $\phi(\mathbf{y}, \mathbf{z}) = \mathbf{y}^T \mathbf{M}(\phi) \mathbf{z}$. A symmetric bilinear form ϕ on a vector space of dimension d, is said to be *non-degenerate* if the rank of $\mathbf{M}(\phi)$ w.r.t to some basis is equal to d.

Example: For the $d-$ dimensional euclidean space with standard basis and ϕ as dot product we get $\mathbf{M}(\phi) = \mathbf{I}_d$

Definition D.6.4. (**Pseudo-euclidean Spaces**) A real vector space $\mathbb{R}^{d^+, d^-}$ of dimension $d = d^+ + d^-$, equipped with a non-degenerate symmetric bilinear form ϕ is called a pseudo-euclidean (or Minkowski) vector space of signature (d^+, d^-) if the matrix of ϕ w.r.t a basis $\{\mathbf{b}_i\}_{i=1}^d$

of $\mathbb{R}^{d^+,d^-}$, is given as,

$$\mathbf{M}(\phi) = \begin{pmatrix} \mathbf{I}_{d^+} & \mathbf{0} \\ \mathbf{0} & -\mathbf{I}_{d^-} \end{pmatrix}_{d \times d}$$

Embedding Algorithms The tool that ensures we can produce isometric embeddings is the following result:

Proposition 3. (Goldfarb (1985)) Let $\mathcal{Y} = \{y_0, \dots y_k\}$ be a finite pseudo-metric space equipped with distance function $d_{\mathcal{Y}}$, and let $\mathbf{V} = \{\mathbf{v}_i, \dots, \mathbf{v}_k\}$ be a collection of vectors in $\mathbb{R}^{d^+,d^-}$. Then $\mathcal{Y}$ is isometrically embeddable in $\mathbb{R}^{d^+,d^-}$ if and only if,

$$\langle \mathbf{v}_i, \mathbf{v}_j \rangle_\phi = \frac{1}{2}\Big(d^2_{\mathcal{Y}}(y_i, y_0) + d^2_{\mathcal{Y}}(y_j, y_0) - d^2_{\mathcal{Y}}(y_i, y_j)\Big) \quad \forall i, j \in [k] \tag{D.16}$$

This bilinear form is very similar to the one used for MDS embeddings (Kruskal and Wish, 1978)—it is closely related to the squared distance matrix. The main information needed is what the signature (i.e., how many positive, negative, and zero eigenvalues) of this bilinear form is. If the dimension of the pseudo-Euclidean space we choose to embed in is at least as large as the number of positive and negative eigenvalues, we can obtain isometric embeddings. Because we are working with finite metric spaces, this number is always finite, and, in fact, is never larger than the size of the metric space. This means we can always produce isometric embeddings.

The practical aspects of how to produce the embedding are shown in the first half of Algorithm 3. The basic idea is to do an eigendecomposition and capture eigenvectors corresponding to the positive and negative eigenvalues. These allow us to perfectly reproduce the positive and negative components of the distances separately; the resulting distance is the difference between the two components. The process of performing the eigendecomposition is standard, so that the overall procedure has the same complexity as running MDS. Compare this to MDS: there, we only capture the eigenvectors corresponding to the positive eigenvalues and ignore the negative ones. Otherwise, the procedure is identical.

We note that in fact it is possible to embed pseudo-metric spaces isometrically into pseudo-

Euclidean spaces, but we never use this fact. Our only application of this tool is to embed conventional metric spaces. However, our results directly lift to this more general setting.

The idea of using pseudo-Euclidean spaces for embeddings that can then be used in kernel-based or other classifiers or other approaches to machine learning is not new. For example, Pękalska et al. (2001) used these spaces for kernel-based learning, Laub et al. (2006) used them for generic pairwise learning, and Pękalska et al. (2006) showed that they are among several non-standard spaces that provide high-quality representations. Our contribution is using these in the context of weak supervision and learning latent variable models.

Dimensionality We also give more detail on the example we provided showing that pseudo-Euclidean embeddings can have arbitrarily better dimensionality compared to one-hot encodings. The idea here is simple. We start with a particular kind of tree with a root and three branches that are simply long chains (paths) and have t nodes each, for a total of $3t + 1$ nodes. One-hot encodings have dimension that scales with the number of nodes, i.e., dimension $3t + 1$.

Pseudo-euclidean embeddings enable us to embed such a tree into a space of finite (and in fact, very small) dimension while preserving the shortest-hops distances between each pair of nodes in the graph. As described above, the key question is what the number of positive and negative eigenvalues for the squared distance matrix (and thus the bilinear form) is. Fortunately, for such graphs, the signature of the squared-distance matrix is known (Theorem 20 in Bapat and Sivasubramanian (2016)). Applying this result shows that the pseudo-Euclidean dimension is just 3, a tiny fixed value regardless of the value of t above.

Symbol	Definition
$\mathcal{X}$	feature space
$\mathcal{Y}$	label metric space
$\mathcal{Y}_s$	support of prior distribution on true labels
$d_{\mathcal{Y}}$	label metric (distance) function
$x_1, x_2, \ldots, x_n$	unlabeled datapoints from $\mathcal{X}$
$y_1, y_2, \ldots, y_n$	latent (unobserved) labels from $\mathcal{Y}$
$s_1, s_2, \ldots, s_m$	labeling functions / sources
$\lambda_1, \lambda_2, \ldots, \lambda_m$	output of labeling functions (LFs)
$\boldsymbol{\lambda}_1, \boldsymbol{\lambda}_2, \ldots, \boldsymbol{\lambda}_m$	pseudo-Euclidean embeddings of LFs outputs
$\lambda_{a,i}$	output of ath LF on ith data point x_i
$\boldsymbol{\lambda}_{a,i}$	pseudo-Euclidean embedding of output of ath LF on ith data point x_i
n	number of data points
m	number of LFs
k	size of the support of prior on $\mathcal{Y}$ i.e. $k = \lvert\mathcal{Y}_s\rvert$
r	size of $\mathcal{Y}$ for the finite case
$\theta_a, \hat{\theta}_a$	true and estimated canonical parameters of model in (6.3)
$\boldsymbol{\theta}, \hat{\boldsymbol{\theta}}$	true and estimated canonical parameters arranged as vectors
$\mathbb{E}[d^2_{\mathcal{Y}}(\lambda_a, y)]$	mean parameters in (6.3)
g	pseudo-Euclidean embedding mapping
$\mathbf{P}$	true noise model $P_{ij} = P_{\boldsymbol{\theta}}(\tilde{Y} = y_j \lvert Y = y_i)$ with true parameters $\boldsymbol{\theta}$
$\mathbf{Q}$	estimated noise model with parameters $\hat{\boldsymbol{\theta}}$, $Q_{ij} = P_{\hat{\boldsymbol{\theta}}}(\tilde{Y} = y_j \lvert Y = y_i)$
Λ	a random element in $\mathcal{Y}^m$ the m-fold Cartesian product of $\mathcal{Y}$
$\Lambda^{(u)}$	uth element in $\mathcal{Y}^m$
$\boldsymbol{\mu}^+_{a,y}, \boldsymbol{\mu}^-_{a,y}$	means of distributions in (6.4) corresponding to $\mathbb{R}^{d^+}, \mathbb{R}^{d^-}$
$\epsilon(d^+), \epsilon(d^-)$	error in recovering the mean parameters (6.6)
σ	proxy noise variance in (6.4)
$F(x, y)$	the score function in (6.2) with true labels
$\tilde{F}_p(x, y), \tilde{F}_q(x, y)$	the score function in (6.9) with noisy labels from distributions $\mathbf{P}$ and $\mathbf{Q}$
$\hat{f}$	minimizer of F defined in (6.2)
$\hat{f}_p, \hat{f}_q$	minimizers of $\tilde{F}_p, \tilde{F}_q$ as defined in (6.2)
$\sigma_{\max}(\mathbf{P})$	maximum singular value of $\mathbf{P}$
$\sigma_{\min}(\mathbf{P})$	minimum singular value of $\mathbf{P}$
$\kappa(\mathbf{P})$	the condition number of matrix $\mathbf{P}$
$\omega(\mathbf{u}, \mathbf{v})$	angle between vectors $\mathbf{u}, \mathbf{v} \in \mathbb{R}^d$

Table D.1: Glossary of variables and symbols used in this paper.

Appendix E

Appendix to Chapter 8

The Appendix is organized as follows. We summarize the notation in Table E.1. Then we give the proof of the main theorem (Theorem E.1.6) and the proofs of supporting lemmas in Section E.1.1. Then we provide additional experiments and insights from them in Section E.2.

E.1 Glossary

The notation is summarized in Table E.1 below.

E.1.1 Proofs

Summary of the setting. At each time t, we observe $x_t \overset{i.i.d}{\sim} (1-\gamma)\mathcal{D}_{id} + \gamma\mathcal{D}_{ood}$, and $s_t = g(x_t)$ is the corresponding score. If $s_t \leq \hat{\lambda}_{t-1}$, then it is considered an OOD point and hence gets a human label for it and we get to know whether it is in fact OOD or ID. If $s_t > \hat{\lambda}_{t-1}$, then it is considered an ID point and hence gets a human label only with probability p. So, we get to know whether it is truly ID or not with probability p. Now we have to update the threshold, $\hat{\lambda}_t$, such that the $\text{FPR}(\hat{\lambda}_t) \leq \alpha$ for all t, while trying to maximize $\text{TPR}(\hat{\lambda}_t)$. Our approach is based on constructing an unbiased estimator of $\text{FPR}(\lambda)$ using the OOD samples received till time t and in conjunction with confidence intervals for $\text{FPR}(\lambda)$ for at all thresholds $\lambda \in \Lambda$ that is valid for all times simultaneously. Together, at each time t, these give us a reliable upper bound on the true

Symbol	Definition
$\mathcal{X}$	feature space.
$\mathcal{Y}$	label space, $\{+1, -1\}$, $+1$ for ID and -1 for OOD .
$\mathcal{D}_{id}, \mathcal{D}_{ood}$	distributions of ID and OOD points.
γ	mixing ratio of OOD and ID distributions.
λ	threshold for OOD classification.
$\text{FPR}(\lambda)$	population level false positive rate with threshold λ.
$\text{TPR}(\lambda)$	population level true positive rate with threshold λ.
$\widehat{\text{FPR}}(\lambda, t)$	empirical FPR at time t (see eq. (E.1)).
$\hat{\lambda}_t$	the estimated threshold at round t.
x_t, y_t	sample and the true label at time t .
g	OOD uncertainty quantification (score) function.
$s_u^{(o)}$	score of u^{th} OOD sample.
$i_u^{(o)}$	indicator variable denoting whether $s_u^{(o)}$ was importance sampled or not.
$N_t^{(o)}$	number of OOD points till time t.
$N_t^{(o,p)}$	number of OOD points sampled using importance sampling until time t.
β_t	it is equal to $N_t^{(o,p)}/N_t^{(o)}$.
p	probability for importance sampling.
δ	failure probability.
α	user given upper bound on FPR that the algorithm needs to maintain.
η	the algorithm is in $\eta-$optimality if close $\text{FPR}(\lambda^*) - \text{FPR}(\hat{\lambda}_t) \leq \eta$.
$\Lambda_{\min}, \Lambda_{\max}$	the minimum and maximum scores(thresholds) considered by the algorithm.
ν	discretization parameter for the interval $[\Lambda_{\min}, \Lambda_{\max}]$ set by the user.
$\psi(t, \delta)$	LIL based confidence interval at time t.

Table E.1: Glossary of variables and symbols used in this paper.

$\text{FPR}(\lambda)$ for all λ enabling us to find the smallest λ such that the upper bound on $\text{FPR}(\lambda)$ is at most α. Let $S_t^{(o)} = \{s_1^{(o)}, \dots s_{N_t^{(o)}}^{(o)}\}$ denote the scores of the points that have been truly identified as OOD points from human feedback and $I_t^{(o)}$ be the corresponding time points. We estimate the FPR as follows,

$$\widehat{\text{FPR}}(\lambda, t) = \frac{1}{N_t^{(o)}} \sum_{u \in I_t^{(o)}} Z_u(\lambda), \quad \text{where } Z_u(\lambda) := \begin{cases} \mathbf{1}(s_u^{(o)} > \lambda), & \text{if } s_u^{(o)} \leq \hat{\lambda}_{u-1} \\ \frac{1}{p}\mathbf{1}(s_u^{(o)} > \lambda), & \text{w.p. } p \text{ if } s_u^{(o)} > \hat{\lambda}_{u-1} \\ 0, & \text{w.p. } 1-p \text{ if } s_u^{(o)} > \hat{\lambda}_{u-1} \end{cases} . \tag{E.1}$$

Proof outline. To obtain the guarantees in Theorem E.1.6 we need confidence intervals $\psi(t, \delta)$

that are simultaneously valid with high probability for the FPR estimates at all time points and all thresholds. There is a rich line of work that provides tight confidence intervals valid for all times based on the Law of Iterated Logarithm (LIL) (Khinchine, 1924; Kolmogorov, 1929; Smirnov, 1944). Non-asymptotic versions of LIL have been proved in various settings, e.g., multi-armed bandits (Jamieson et al., 2014) and for quantile estimation (Howard and Ramdas, 2022). Roughly speaking, these bounds provide confidence intervals that are $\mathcal{O}(\sqrt{\log\log(t)/t}$ and are known to be tight. However, most of them assume the samples to be i.i.d. In our setting, our treatment of observing the human feedback is dependent on whether the score is above or below $\hat{\lambda}_{t-1}$, which itself is estimated using all the past data, which creates dependence and prevents us from utilizing results developed for i.i.d. samples (Howard and Ramdas, 2022). The main technical challenge is to show that the upper confidence bound in eq. (8.2) holds for all time and all thresholds with dependent samples used in estimating the FPR in eq. (E.1). We handle this by first showing that there is a martingale structure in our estimated FPR (eq. (E.1)). We then exploit this structure by using LIL results for martingales (Balsubramani, 2015). A limitation of (Balsubramani, 2015) is that it can only provide us confidence intervals valid for the FPR estimate for a given threshold λ. However, we need intervals that are simultaneously valid for all λ as well. Building upon the work (Balsubramani, 2015) we derive confidence intervals that are simultaneously valid for all t and finitely many thresholds. Eq. (8.2) shows the $\psi(t, \delta)$ we obtain.

Next, we show that the above estimator $\widehat{\text{FPR}}(\lambda, t)$ is indeed an unbiased estimator of the false positive rate $\text{FPR}(\lambda)$.

Lemma E.1.1. *Let $p > 0$, $\widehat{FPR}(\lambda, t)$ as defined in eq. (E.1) is an unbiased estimate of the true $FPR(\lambda)$, i.e., $\mathbb{E}[\widehat{FPR}(\lambda, t)] = FPR(\lambda)$.*

Proof. Let $i_t^{(o)}$ be the indicator variable denoting whether $s_t^{(o)}$ was sampled using importance sampling (i.e. $i_t^{(o)} = 1$) or not (i.e. $i_t^{(o)} = 0$). Denote the pair as $r_t^{(o)} = (s_t^{(o)}, i_t^{(o)})$ for brevity. The proof is by induction. Since the first sample is drawn without any importance sampling so, we have $\mathbb{E}_{r_1^{(o)}}[\widehat{\text{FPR}}(\lambda, 1)] = \text{FPR}(\lambda)$. Now assume that $\mathbb{E}_{r_{t-1}^{(o)}, \ldots, r_1^{(o)}}[\widehat{\text{FPR}}(\lambda, t-1)] = \text{FPR}(\lambda)$.

$$
\begin{aligned}
\mathbb{E}_{r_t^{(o)}, r_{t-1}^{(o)}, \ldots, r_1^{(o)}}[\widehat{\mathrm{FPR}}(\lambda, t)] &= \mathbb{E}\Big[\frac{1}{N_t^{(o)}} \sum_{u \in I_t^{(o)}} Z_u(\lambda)\Big] \\
&= \mathbb{E}\Big[\frac{Z_t(\lambda)}{N_t^{(o)}} + \frac{N_t^{(o)} - 1}{N_t^{(o)}} \frac{1}{N_t^{(o)} - 1} \sum_{u \in I_{t-1}^{(o)}} Z_u(\lambda)\Big] \\
&= \mathbb{E}\Big[\frac{Z_t(\lambda)}{N_t^{(o)}} + \frac{N_t^{(o)} - 1}{N_t^{(o)}} \widehat{\mathrm{FPR}}(\lambda, t-1)\Big] \\
&= \frac{1}{N_t^{(o)}}\Big[\mathbb{E}[Z_t(\lambda)] + (N_t^{(o)} - 1) \cdot \mathbb{E}[\widehat{\mathrm{FPR}}(\lambda, t-1)]\Big] \\
&= \frac{1}{N_t^{(o)}}\Big[\mathbb{E}[Z_t(\lambda)] + (N_t^{(o)} - 1) \cdot \mathrm{FPR}(\lambda)\Big] \\
&= \frac{1}{N_t^{(o)}}\Big[\mathbb{E}_{r_t^{(o)} | \hat{\lambda}_{t-1}}[Z_t(\lambda)] + (N_t^{(o)} - 1) \cdot \mathrm{FPR}(\lambda)\Big] \\
&= \frac{1}{N_t^{(o)}}\Big[\mathrm{FPR}(\lambda) + (N_t^{(o)} - 1) \cdot \mathrm{FPR}(\lambda)\Big] \\
&= \mathrm{FPR}(\lambda)
\end{aligned}
$$

□

Having an unbiased estimator solves one part of the problem. In addition, we need confidence intervals on this estimate that are valid for anytime and for the choices of λ considered. Due to the dependence between the samples, we cannot directly apply similar results developed for quantile estimation in the i.i.d. setting (Howard and Ramdas, 2022). Fortunately, part of this problem has been addressed in (Balsubramani, 2015), where they provide anytime valid confidence intervals when the estimators form a martingale sequence. We restate this result in the following lemma E.1.2 and then, building upon this result, in the next lemma E.1.3, we derive such confidence intervals for our setting.

Lemma E.1.2. *((Balsubramani, 2015)) Let $\overline{M}_t$ be a martingale and suppose $|\overline{M}_t - \overline{M}_{t-1}| \leq \rho_t$ for constants $\{\rho_t\}_{t>1}$, let $m_0 = \min_{t \geq 1} |\overline{M}_t|$. Fix any $\delta \in (0,1)$, and let $t_0 = \min\{u : \sum_{t=1}^{u} \rho_t^2 \geq 173 \log(\frac{4}{\delta})\}$ then,*

$$\mathbb{P}\Bigg(\exists t \geq t_0 : |\overline{M}_t| \geq \sqrt{3\Big(\sum_{i=1}^{t}\rho_i^2\Big)\Big(2\log\log\big(\frac{3\sum_{i=1}^{t}\rho_i^2}{m_0}\big)+\log\big(\frac{2}{\delta}\big)\Big)}\Bigg) \leq \delta \tag{E.2}$$

Proof. This lemma is a restatement of Theorem 4 in (Balsubramani, 2015). For proof details, please see (Balsubramani, 2015). □

In the next lemma, we show that the sums of $Z_u(\lambda)$ form a martingale sequence, allowing us to apply the results from the above lemma (E.1.2), and then we generalize it to all λ in some finite set.

Lemma E.1.3. *(Anytime valid confidence intervals on FPR) Let $X_t^{(o)} = \{x_1^{(o)}, \dots x_{N_t^{(o)}}^{(o)}\}$ be the samples drawn from D_{ood} till round t and let $S_t^{(o)} = \{s_1^{(o)}, \dots s_{N_t^{(o)}}^{(o)}\}$ be the scores of these points, let $c_t = 1 - \beta_t + \frac{\beta_t}{p^2}$, $\beta_t = \frac{N_t^{(o,p)}}{N_t^{(o)}}$ and $N_t^{(o,p)}$ is the number of points sampled using importance sampling until time t and $\nu \in (0,1)$ is a discretization parameter set by the user. Let $\Lambda = \{\Lambda_{\min}, \Lambda_{\min}+\nu, \dots, \Lambda_{\max}\}$. Let $n_0 = \min\{u : c_u N_u^{(o)} \geq 173\log(\frac{4}{\delta})\}$ and t_0 be such that $N_{t_0}^{(o)} \geq n_0$. , then for any $\delta \in (0,1)$,*

$$\mathbb{P}\Bigg(\exists t \geq t_0 : \sup_{\lambda\in\Lambda} \widehat{FPR}(\lambda,t) - FPR(\lambda) \geq \psi(t,\delta)\Bigg) \leq \delta \tag{E.3}$$

for,

$$\psi(t,\delta) = \sqrt{\frac{3c_t}{N_t^{(o)}}\left[2\log\log\left(\frac{3c_t N_t^{(o)}}{2}\right)+\log\left(\frac{2|\Lambda|}{\delta}\right)\right]} \tag{E.4}$$

Proof. First, we show that we have a martingale sequence as follows,

Let $M_t(\lambda) = \sum_{u=1}^{N_t^{(o)}} Z_u(\lambda)$, and consider the centered random variables,

$$\overline{M}_t(\lambda) = M_t(\lambda) - \mathbb{E}[M_t(\lambda)] \quad \text{and} \quad \overline{Z}_t(\lambda) = Z_t(\lambda) - \text{FPR}(\lambda)$$

Let $\mathcal{F}_t$ be the $\sigma-$algebra of events till time t i.e. $(s_1^{(o)}, i_1^{(o)}), \dots, (s_{t-1}^{(o)}, i_{t-1}^{(o)}), (s_t^{(o)}, i_t^{(o)})$.

It is easy to see that $\mathbb{E}[\overline{M}_t] \leq \frac{1}{p} < \infty$ and $\overline{M}_t$ is $\mathcal{F}_t$-measurable for all $t > 1$. Further, we can see,

$$\mathbb{E}\big[\overline{M}_t(\lambda)|\mathcal{F}_{t-1}\big] = \mathbb{E}[\overline{Z}_t(\lambda) + \overline{M}_{t-1}(\lambda)|\mathcal{F}_{t-1}] = \mathbb{E}[\overline{Z}_t(\lambda)|\mathcal{F}_{t-1}] + \mathbb{E}[\overline{M}_{t-1}(\lambda)|\mathcal{F}_{t-1}] = \overline{M}_{t-1}(\lambda)$$

Since, $\mathbb{E}[\overline{Z}_t(\lambda)|\mathcal{F}_{t-1}] = 0$ and $\mathbb{E}[\overline{M}_{t-1}(\lambda)|\mathcal{F}_{t-1}] = \overline{M}_{t-1}(\lambda)$. Thus, we have that $\overline{M}_t$ is a martingale sequence. Further, we also have the following,

$$|\overline{M}_t(\lambda) - \overline{M}_{t-1}(\lambda)| \leq \begin{cases} 1 & \text{if } i_t^{(o)} = 0 \\ \frac{1}{p} & \text{if } i_t^{(o)} = 1 \end{cases}$$

Let $\beta_t \in (0, 1)$ be the fraction of OOD points sampled using probability p till round t. Let $N_t^{(o)}$ be the total number of points OOD points sampled till round t and $N_t^{(o,p)}$ be the points sampled from importance sampling, then $\beta_t = \frac{N_t^{(o,p)}}{N_t^{(o)}}$.

Let $c_t = 1 - \beta_t + \frac{\beta_t}{p^2}$. We know p and the number of points sampled with importance sampling, without importance sampling, we know β_t, c_t are at time t. Applying lemma E.1.2 we get the following result for a given λ,

$$\mathbb{P}\left(\exists t \geq t_0 : \overline{M}_t(\lambda) \geq \sqrt{3\left(c_t N_t^{(o)}\right)\left(2\log\log\left(3c_t N_t^{(o)}\right) + \log\left(\frac{2}{\delta}\right)\right)}\right) \leq \delta \tag{E.5}$$

$$\mathbb{P}\left(\exists t \geq t_0 : \widehat{\mathrm{FPR}}(\lambda, t) - \mathrm{FPR}(\lambda, t) \geq \sqrt{\frac{3c_t}{N_t^{(o)}}\left(2\log\log\left(3c_t N_t^{(o)}\right) + \log\left(\frac{2}{\delta}\right)\right)}\right) \leq \delta \tag{E.6}$$

Doing the union bound for the failure probability over all $\lambda \in \Lambda$, (where $|\Lambda| < \infty$) gives us the result.

□

Our performance guarantees in the main theorem E.1.6 are based on $\psi(t, \delta)$ becoming smaller than certain values. In the next lemma, we derive a bound on $N_t^{(o)}$ such that $\psi(t, \delta)$ is at most μ and use it in the proof of the main theorem E.1.6.

Lemma E.1.4. *Let* $\psi(t,\delta) = \sqrt{\frac{3c_t}{N_t^{(o)}}\left(2\log\log\left(3c_tN_t^{(o)}\right) + \log\left(\frac{2|\Lambda|}{\delta}\right)\right)}$, *and let there be a constant* C_0 *and time* T_0, *such that* $\beta_t \leq C_0p^2$ *for all* $t \geq T_0$ *(worst case* $T_0 = 1$ *and* $C_0 = 1/p^2$*). Then* $\psi(t,\delta) \leq \mu$ *for any* $t > T_\mu > T_0$ *such that* $N_{T_\mu}^{(o)} = \frac{10(C_0+1)}{\mu^2}\log\left(\frac{|\Lambda|}{\delta}\log(\frac{5(C_0+1)}{\mu})\right)$.

Proof. First we write a simplified form of $\psi(t,\delta)$ for all $t > T_0$ as follows,

$$\psi(t,\delta) = \sqrt{\frac{3(C_0+1)}{N_t^{(o)}}\left(2\log\log\left(3(C_0+1)N_t^{(o)}\right) + \log\left(\frac{2|\Lambda|}{\delta}\right)\right)}$$

In the above equation we used the bound on $\beta_t \leq C_0p^2$ in the equation $c_t = 1 - \beta_t + \beta_t/p^2$ leading to $c_t \leq C_0 + 1$, Now, for brevity let $a_1 = 3(C_0+1)$ and $a_2 = 2|\Lambda|$ and rewrite $\psi(t,\delta)$ as follows,

$$\psi^2(t,\delta) = \frac{a_1}{N_t^{(o)}}\left(2\log\log\left(a_1N_t^{(o)}\right) + \log\left(\frac{a_2}{\delta}\right)\right) \leq \frac{2a_1}{N_t^{(o)}}\left(\log\left(\frac{a_2}{\delta}\log\left(a_1N_t^{(o)}\right)\right)\right)$$

We want to find $N_t^{(o)}$ such that $\psi^2(t,\delta) \leq \mu^2$. It is difficult to directly invert this function. To get a bound on $N_t^{(o)}$ we first assume the following form for it with unknown constants $b_1, b_2, b_3 > 0$ and then figure out the constants by simplifying $\psi^2(N_t^{(o)})$ and constraining it to be at most μ^2.

$$\text{Let } N_{T_\mu}^{(o)} = \frac{b_1}{\mu^2}\log\left(\frac{a_2}{b_3\delta}\log\left(\frac{b_2}{\mu}\right)\right)$$

$$\begin{aligned}
\psi^2(T_\mu,\delta) &\leq \frac{2a_1}{N_{T_\mu}^{(o)}}\log\left[\frac{a_2}{\delta}\log(a_1N_{T_\mu}^{(o)})\right] \\
&= \frac{2a_1\mu^2}{b_1\log\left(\frac{a_2}{b_3\delta}\log\left(\frac{b_2}{\mu}\right)\right)}\log\left[\frac{a_2}{\delta}\log\left\{\frac{a_1b_1}{\mu^2}\log\left(\frac{a_2}{b_3\delta}\log\left(\frac{b_2}{\mu}\right)\right)\right\}\right] \\
&\overset{(i)}{\leq} \frac{2a_1\mu^2}{b_1\log\left(\frac{a_2}{b_3\delta}\log\left(\frac{b_2}{\mu}\right)\right)}\log\left[\frac{a_2}{\delta}\log\left\{\frac{a_1b_1}{\mu^2}\log\left(\frac{a_2}{b_3\delta}\frac{b_2}{\mu}\right)\right\}\right] \\
&= \frac{2a_1\mu^2}{b_1\log\left(\frac{a_2}{b_3\delta}\log\left(\frac{b_2}{\mu}\right)\right)}\log\left[\frac{a_2}{\delta}\log\left\{\frac{a_1b_1}{\mu^2}\log\left(\frac{a_2b_2}{b_3\delta\mu}\right)\right\}\right]
\end{aligned}$$

$$
\begin{aligned}
&\overset{(ii)}{\leq} \frac{2a_1\mu^2}{b_1 \log\left(\frac{a_2}{b_3\delta}\log\left(\frac{b_2}{\mu}\right)\right)} \log\left[\frac{a_2}{\delta}\log\left\{\frac{a_1b_1}{\mu^2}\frac{a_2b_2}{b_3\delta\mu}\right\}\right] \\
&= \frac{2a_1\mu^2}{b_1 \log\left(\frac{a_2}{b_3\delta}\log\left(\frac{b_2}{\mu}\right)\right)} \log\left[\frac{a_2}{\delta}\log\left\{\frac{a_1b_1}{\mu^2}\frac{a_2b_2}{b_3\delta\mu}\right\}\right] \\
&= \frac{2a_1\mu^2}{b_1 \log\left(\frac{a_2}{b_3\delta}\log\left(\frac{b_2}{\mu}\right)\right)} \log\left[\frac{a_2}{\delta}\log\left\{\frac{a_1b_1a_2}{b_3b_2^2\delta}\left(\frac{b_2}{\mu}\right)^3\right\}\right] \\
&\overset{(iii)}{\leq} \frac{2a_1\mu^2}{b_1 \log\left(\frac{a_2}{b_3\delta}\log\left(\frac{b_2}{\mu}\right)\right)} \log\left[\frac{a_2}{\delta}\frac{a_1b_1a_2}{b_3b_2^2\delta}\log\left\{\left(\frac{b_2}{\mu}\right)^3\right\}\right] \\
&= \frac{2a_1\mu^2}{b_1 \log\left(\frac{a_2}{b_3\delta}\log\left(\frac{b_2}{\mu}\right)\right)} \log\left[\frac{a_2}{\delta}\frac{3a_1b_1a_2}{b_3b_2^2\delta}\log\left(\frac{b_2}{\mu}\right)\right] \\
&= \frac{2a_1\mu^2}{b_1 \log\left(\frac{a_2}{b_3\delta}\log\left(\frac{b_2}{\mu}\right)\right)} \log\left[\left(\frac{a_2}{b_3\delta}\right)^2\frac{3a_1b_1b_3}{b_2^2}\log\left(\frac{b_2}{\mu}\right)\right] \\
&\overset{(iv)}{\leq} \frac{2a_1\mu^{2\frac{3a_1b_1b_3}{b_2^2}}}{b_1 \log\left(\frac{a_2}{b_3\delta}\log\left(\frac{b_2}{\mu}\right)\right)} \log\left[\left(\frac{a_2}{b_3\delta}\right)^2\log\left(\frac{b_2}{\mu}\right)\right] \\
&\overset{(v)}{\leq} \frac{2a_1\mu^2\cdot 2^{\frac{3a_1b_1b_3}{b_2^2}}}{b_1 \log\left(\frac{a_2}{b_3\delta}\log\left(\frac{b_2}{\mu}\right)\right)} \log\left[\frac{a_2}{b_3\delta}\log\left(\frac{b_2}{\mu}\right)\right] \\
&= \frac{12\mu^2a_1^2b_3}{b_2^2}.
\end{aligned}
$$

The inequalities $(i), (ii)$ follow from $\log(x) \leq x$ for any $x > 0$.

The inequality (iii) comes from $\log(ax) \leq a\log(x)$ for any $a > 2, x > 2$. We use $a = \frac{a_1b_1a_2}{b_3b_2^2\delta}$ and $x = \left(\frac{b_2}{\mu}\right)^3$, this enforces the following constraints,

$$\frac{b_2}{\mu} > 2^{1/3} \tag{E.7}$$

$$\frac{a_1b_1a_2}{b_3b_2^2\delta} > 2 \tag{E.8}$$

For (iv) we again use $\log(ax) \leq a\log(x)$ with $a = \frac{3a_1b_1b_3}{b_2^2}$ and $x = \left(\frac{a_2}{b_3\delta}\right)^2\log\left(\frac{b_2}{\mu}\right)$, this enforces the following constraints,

$$\frac{3a_1b_1b_3}{b_2^2} > 2 \tag{E.9}$$

$$(\frac{a_2}{b_3\delta})^2 \log(\frac{b_2}{\mu}) > 2 \tag{E.10}$$

Lastly, (v) follows by using $\log(x^a y) \leq a \log(xy)$ for any $x > 0, a > 1, y > 1$. For this we use $x = \frac{a_2}{b_3\delta}$ and $y = \log(\frac{b_2}{\mu})$, leading the following constraints,

$$\log(\frac{b_2}{\mu}) > 1 \tag{E.11}$$

For $\psi^2(T_\mu) \leq \mu^2$, we need

$$12a_1^2 b_3 \leq b_2^2 \tag{E.12}$$

Let $b_3 = 2, b_1 = 10a_1, b_2 = 5a_1$ then the constraints E.7,E.8,E.9,E.10,E.11 and E.12 are satisfied (when $|\Lambda| \geq 10$) for any $\mu \in (0,1), \delta \in (0,1)$. Thus we have,

$$\psi(T_\mu, \delta) \leq \mu \text{ for } N_{T_\mu} = \frac{10(C_0+1)}{\mu^2} \log\Big(\frac{|\Lambda|}{\delta} \log(\frac{5(C_0+1)}{\mu})\Big) \tag{E.13}$$

□

Lemma E.1.5. *Let the data points $\{x_t\}_{t\geq 1}$ be independent draws from the mixture distribution $(1-\gamma)\mathcal{D}_{id}+\gamma\mathcal{D}_{ood}$, and $N_t^{(o)}$ be the number of OOD points received till time t from this distribution, then for any $\delta \in (0,1)$ for any $t \geq T_k$ we have $N_t^{(o)} \geq k$ w.p. $1-\delta$, where T_k is given as follows,*

$$T_k = \frac{2k}{\gamma} + \frac{1}{\gamma^2} \log(\frac{1}{\delta}). \tag{E.14}$$

Proof. We want to find t such that $N_t^{(o)} \geq k$ w.p. $1-\delta$. This is the same as finding the number of coin tosses of a coin with bias γ so that the number of heads observed is at least k. Applying Hoeffding's inequality gives us the following w.p. $1-\delta$,

$$N_t^{(o)} \geq \gamma t - \sqrt{\frac{t}{2} \log(\frac{1}{\delta})}.$$

Equating the r.h.s. above with k and solving for t will give us the desired bound on t. Note that

it is enough to have an upper bound on t that satisfies the following and then use that upper bound as T_k.

$$\gamma t - \sqrt{\frac{t}{2}\log(\frac{1}{\delta})} = k.$$

To simplify the calculations, let $c = \sqrt{\frac{1}{2}\log\frac{1}{\delta}}$ and let $t = u^2$ then we have the following quadratic equation,

$$\gamma u^2 - cu - k = 0.$$

Considering the larger of the two solutions,

$$u = \frac{c + \sqrt{c^2 + 4k\gamma}}{2\gamma}.$$

Using the fact that for any $a, b \geq 0$, $\sqrt{a+b} \leq \sqrt{a} + \sqrt{b}$,

$$u \leq \frac{c + \sqrt{c^2} + \sqrt{4k\gamma}}{2\gamma} = \frac{2c + 2\sqrt{k\gamma}}{2\gamma} = \frac{c}{\gamma} + \sqrt{\frac{k}{\gamma}}.$$

Lastly, using $(a+b)^2 \leq 2a^2 + 2b^2$ for any $a, b \in \mathbb{R}$ we get the following upper bound on t,

$$t = u^2 \leq \frac{2c^2}{\gamma^2} + \frac{2k}{\gamma} = \frac{2k}{\gamma} + \frac{1}{\gamma^2}\log(\frac{1}{\delta}).$$

□

Theorem E.1.6. *Let $\alpha, \delta, p, \gamma \in (0,1)$. Let $x_t \overset{i.i.d}{\sim} (1-\gamma)\mathcal{D}_{id} + \gamma\mathcal{D}_{ood}$ and let $c_t = 1 - \beta_t + \frac{\beta_t}{p^2}$, $\beta_t = \frac{N_t^{(o,p)}}{N_t^{(o)}}$ where $N_t^{(o,p)}$ is the number of OOD points sampled using importance sampling until time t and $N_t^{(o)}$ is the total number of OOD points observed till time t. Let $n_0 = \min\{u : c_u N_u^{(o)} \geq 173\log(\frac{8}{\delta})\}$ and t_0 be such that $N_{t_0}^{(o)} \geq n_0$. If Algorithm 4 uses the optimization problem (P2) to find the thresholds with the upper confidence term $\psi(N_t^{(o)}, \delta/2)$ given by eq. (8.2), then there exist constants $C_1, C_2, C_3 > 0$ such that with probability at least $1-\delta$,*

1. ***Controlled FPR.*** *For all $t \geq t_0$, $FPR(\hat{\lambda}_t) \leq \alpha$.*

2. ***Time to reach feasibility.*** *The algorithm will find a feasible threshold, $\hat{\lambda}_t$ such that $\widehat{FPR}(\hat{\lambda}_t)+$*

$\psi(N_t^{(o)}) \leq \alpha$, *for all* $t \geq \max(t_0, T_f)$, *where,* $T_f = \frac{2C_1}{\gamma\alpha^2}\log\left(\frac{4C_2}{\delta}\log(\frac{C_3}{\alpha})\right) + \frac{1}{\gamma^2}\log(\frac{4}{\delta})$.

3. ***Time to reach*** $\eta-$***optimality.*** *For all* $t \geq \max(t_0, T_{\eta\text{-}opt})$, $\hat{\lambda}_t$ *satisfy the* η*-optimality condition in definition 8.3.1, when* $\widehat{FPR}(\hat{\lambda}_{T_{\eta\text{-}opt}}) \in [\alpha - \eta/2, \alpha]$ *and* $T_{\eta\text{-}opt} = \frac{8C_1}{\gamma\eta^2}\log\left(\frac{4C_2}{\delta}\log(\frac{2C_3}{\eta})\right) + \frac{1}{\gamma^2}\log(\frac{4}{\delta})$.

Proof. To prove this, we first obtain confidence intervals on FPR valid w.p. $1-\delta/2$ using Lemma E.1.3. Then applying Lemma E.1.4 on these confidence intervals gives us the number of OOD samples that are sufficient to guarantee a certain width of the confidence intervals and lastly we use Lemma E.1.5 to bound the time point such that we observe a certain number of OOD points till that time. We do this for the second and third points separately each time invoking Lemma E.1.5 with failure probability $\delta/4$ and then union bound over them.

Controlled FPR: This follows from Lemma E.1.3 (with probability $1-\frac{\delta}{2}$) and the fact that the algorithm uses confidence intervals on FPR estimates that are valid for all $t \geq t_0$ for the choices of λ it considers.

Time to reach feasibility: Applying Lemma E.1.4 with $\mu = \alpha$ gives bound on N_{T_f} with $C_1 = 10(C_0+1), C_2 = |\Lambda|, C_3 = 5(C_0+1)$. Then using Lemma E.1.5 with $k = N_{T_f}$ gives us the desired T_f.

Time to reach η*-optimality :* We know, $\text{FPR}(\lambda^*) = \alpha$, and it is given that $\widehat{\text{FPR}}(\hat{\lambda}_t) \in [\text{FPR}(\lambda^*) - \eta/2, \alpha]$

$$\text{FPR}(\hat{\lambda}_t) \in [\text{FPR}(\lambda^*) - \eta/2 - \psi(t,\delta), \alpha]$$

this means $\text{FPR}(\hat{\lambda}_t) \geq \text{FPR}(\lambda^*) - \eta/2 - \psi(t,\delta)$

$$\text{FPR}(\lambda^*) - \text{FPR}(\hat{\lambda}_t) \leq \eta/2 + \psi(t,\delta)$$

If $\psi(t,\delta) \leq \eta/2$ we have, $\text{FPR}(\lambda^*) - \text{FPR}(\hat{\lambda}_t) \leq \eta$. Thus, applying we want to find t for which $\psi(t,\delta) = \eta/2$. Applying lemma E.1.4 with $\mu = \eta/2$ gives bound on N_{T_f} with $C_1 = 40(C_0+1), C_2 = |\Lambda|, C_3 = 10(C_0+1)$. Then using Lemma E.1.5 with $k = N_{T_{opt}}$ gives us the

desired T_{opt}. □

This concludes the proofs of the main results. Next, we present details of the procedure used to solve the optimization problem P2 and additional experiments on synthetic and real datasets.

E.1.2 Additional Details of the Algorithm

We use the following algorithm (Algorithm 12) based on binary search to solve the optimization problem P2, i.e., find $\hat{\lambda}_t$. In addition to the best estimate of current threshold $\hat{\lambda}_t$, it also returns a flag $feasible$ that indicates whether the procedure found a threshold satisfying the constraint in P2 or not.

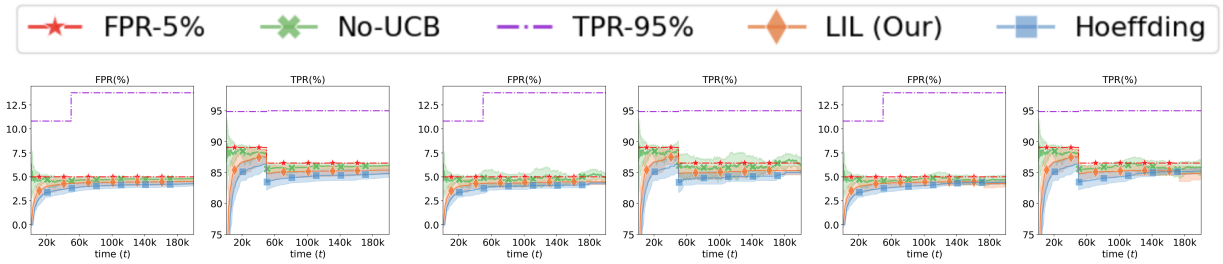
Algorithm 12: SolveOptForLambda

Input: FPR threshold α , S_t

$low = 1, high = \frac{\Lambda_{\max} - \Lambda_{\min}}{\nu}, feasible = False$
while $low < high$ **do**
  $mid = \lceil (low + high)/2 \rceil$
  $\lambda_{\text{mid}} = \Lambda_{\min} + k\nu$
  if $\widehat{\text{FPR}}(\lambda_{\text{mid}}, t) + \psi(t, \delta) \leq \alpha$ **then**
    $feasible = True$
    $high = mid$
  else
    $low = mid$
  end if
end while
Output $feasible, \lambda_{\text{mid}}$

E.2 Additional Experiments and Details

In the simulations we study the effect of changing γ, using different window sizes and the case when the In-Distribution shifts. For the real data experiments, we study the performance of the methods under different settings with different scoring functions on CIFAR-10 and CIFAR-100 as In-Distribution datasets.



(a) No window. (b) 5k window. (c) 10k window.

Figure E.1: Changing ID distribution in synthetic data.

E.2.1 Searching for Constants in LIL-Heuristic

The theoretical LIL bound in eq. (8.2) has constants that can be pessimistic in practice. We get around this by using a LIL-Heuristic bound which has the same form as in eq. (8.2) but with different constants in particular we consider the form in eq. (LIL-Heuristic). We find the constants C_1, C_2, C_3 using a simulation on estimating the bias of a coin with different constants and picking the ones so that the observed failure probability is below 5%.

$$\tilde{\psi}(t,\delta) = C_1\sqrt{\frac{c_t}{N_t^{(o)}}\left(\log\log\left(C_2 c_t N_t^{(o)}\right) + \log\left(\frac{C_3}{\delta}\right)\right)}. \qquad \text{(LIL-Heuristic)}$$

Specifically, we keep $C_3 = 1$, and run for $\delta \in \{0.01, 0.05, 0.1, 0.2, 0.3, 0.4\}$ with varying C_1 from 0.1 to 0.9 and C_2 from 1.5 to 4.75. For each choice of δ, C_1, C_2, we toss an unbiased coin (mean $p = 0.5$) for $T = 10k$ times. For each choice of $t = 1, 2, \cdots, T$, we compute the empirical mean $\hat{p}$ of the coin and define it as a failure if $p \notin [\hat{p} - \tilde{\psi}(t,\delta), \hat{p} + \tilde{\psi}(t,\delta)]$. We run this process for 100 times and compute the average failure probability for each choice of $t = 1, 2, \cdots, T$. We then pick the constant so that the observed average probability is below 5%. Throughout the paper, we use $C_1 = 0.5$ and $C_2 = 0.75$.

E.2.2 Additional Experiments on Synthetic Data

In-Distribution shift : We study the scenario when the ID distribution changes and the OOD distribution remains fixed. In this setting the FPR for any threshold does not change since the OOD does not change but the TPR changes due to the change in ID distribution. Thus, we

expect that with this type of change our method will not violate FPR constraint and it will gradually adapt to the threshold achieving the new TPR.

We simulate the OOD and ID scores using a mixture of two Gaussians $\mathcal{N}_{id}(\mu = 5.5, \sigma = 4)$ and $\mathcal{N}_{ood}(\mu = -5, \sigma = 4)$ with $\gamma = 0.2$. To simulate distribution change we change the ID distribution to $\mathcal{N}_{id}(\mu = 5, \sigma = 4)$ at time $t = 50k$. We run the methods 10 times with different random seeds. The results are shown in Figure E.1. We can clearly see that changing ID distribution(ID scores getting closer to the OOD scores) leads to a decrease in the TPR at the threshold with 5% FPR. Since the estimation of threshold only depends on the FPR estimates and hence only on OOD samples, changing ID distribution does not affect this estimation so the methods perform the same as in the setting of no-distribution shift but get a reduction in the TPR at FPR 5%.

E.2.3 Additional Real OOD Datasets Experiments

We run our proposed system on real ID and OOD datasets with various OOD scoring methods. We use $\alpha = 0.05$, $\delta = 0.2$, and importance sampling probability $p = 0.2$ through all the experiments. we used an Nvidia RTX 6000 graphics processing unit (GPU) to facilitate the inference process and retrieve confidence scores for various datasets. The primary experiments conducted were executed without the GPU.

ID and OOD datasets. We use CIFAR-10 or CIFAR-100 as ID datasets. In the distribution shift setting, if not specified, we use MNIST, SVHN, and Texture as the first mixture of OOD datasets, and TinyImageNet, Places365, and CIFAR-10/100 as the second mixture of OOD datasets by default. We use a pre-trained Resnet-50 model for SSD method, and Resnet-18 for the rest of the methods.

Scoring functions. We use the following scoring functions,

1. **ODIN.** ODIN (Liang et al., 2017) takes the soft-max score from DNNs and scales the score with temperature. A gradient-based input perturbation is also used for better performance. We choose temperature 1000 and input perturbation noise 0.0014, as discussed in (Liang et al., 2017). The results with this scoring function on CIFAR-10 and CIFAR-100 ID data settings

are shown in Figures E.6 and E.13 respectively.

2. **Mahalanobis Distance (MDS).** For a given point x, the Mahalanobis Distance (MDS) based score is its MD from the closest class conditional distribution. We use the MD-based score as given in (Lee et al., 2018a) for detecting OOD and adversarial samples. They compute the scores using representations from various layers of DNNs and combine them to get a better scoring function. We choose input perturbation noise to be 0.0014. The results with this scoring function on CIFAR-10 and CIFAR-100 ID data settings are shown in Figures E.4 and E.10 respectively.

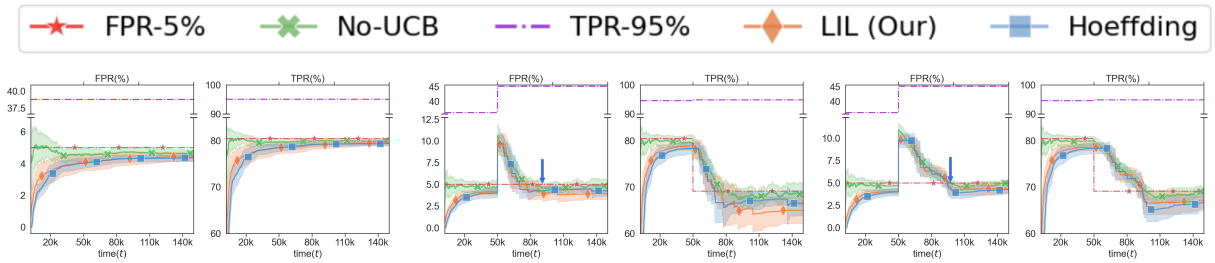
3. **Energy Score (EBO).** This score was proposed in (Liu et al., 2020) and it is well aligned with the probability density of the samples, with low energy implying ID and high energy implying OOD. We choose the temperature parameter to be 1. The results with the EBO scoring function on CIFAR-10 and CIFAR-100 ID data settings are shown in Figures E.3 and E.9 respectively.

4. **SSD**. It is based on computing the Mahalanobis distance in the feature space of the model trained on the unlabeled in-distribution data using self-supervised learning. We use the official implementation of (Sehwag et al., 2021). For CIFAR-10, we use the pre-train model they released. For CIFAR-100, We train a Resnet-50 using a contrastive self-supervised learning loss, SimCLR (Chen et al., 2020). When calculating the distance-based OOD scores, we use one unsupervised clustering center as the only center for ID distribution for both CIFAR-10 and CIFAR-100. The results with this scoring function on CIFAR-10 and CIFAR-100 ID data setting are shown in Figures E.7 and E.12 respectively.

5. **Virtual-logit Match**. Virtual-logit Match (VIM) (Wang et al., 2022c) combines the class-agnostic score from feature space and ID class-dependent logits. Specifically, an additional logit representing the virtual OOD class is generated from the residual of the feature against the principal space and then matched with the original logits by a constant scaling. We set the dimension of the principal space $D = 100$. The results with the VIM scoring function on CIFAR-10 and CIFAR-100 ID data settings are shown in Figures E.5 and E.11 respectively.

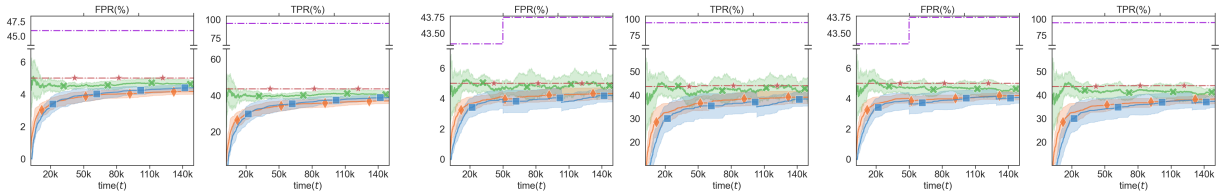
6. **K-Nearest-Neighborhood**. Unlike other methods that impose a strong distributional assumption of the underlying feature space, the KNN-based method (Sun et al., 2022) explores the efficacy of non-parametric nearest-neighbor distance for OOD detection. The distance between the test sample and its k-nearest training IN sample will be used as the score for a threshold based OOD detection. We choose neighbor number $k = 50$. The results with this scoring function on CIFAR-10 and CIFAR-100 ID data settings are shown in Figures E.2 and E.8 respectively.

We also provide visualizations showing the distributions of the scores obtained using these scoring functions. Please see Figures E.14 to E.25.



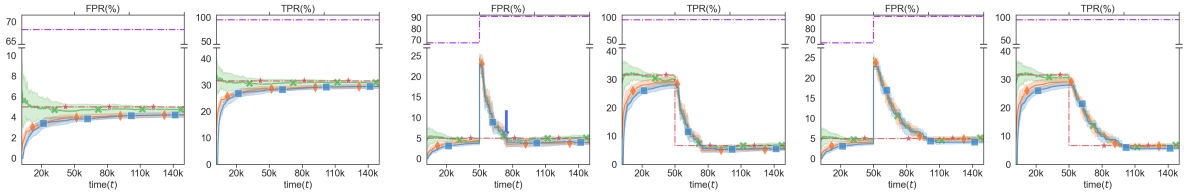
(a) No distribution shift, no window. (b) Distribution shift, 5k window. (c) Distribution shift, 10k window.

Figure E.2: Results with the KNN scores on CIFAR-10 as ID dataset. For (b) and (c) the distribution shifts at $t = 50k$. The arrow indicates the time at which the mean FPR + std. deviation over 10 runs goes below 5% for the LIL method.



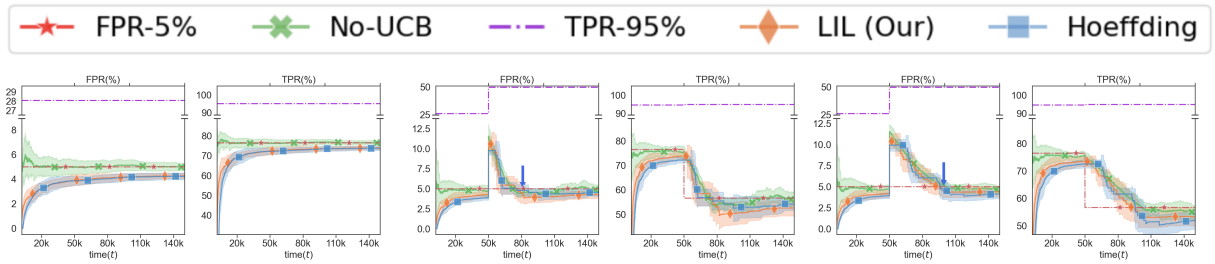
(a) No distribution shift, no window. (b) Distribution shift, 5k window. (c) Distribution shift, 10k window.

Figure E.3: Results with the EBO scores on CIFAR-10 as ID dataset. For (b) and (c) the distribution shifts at $t = 50k$. The arrow indicates the time at which the mean FPR + std. deviation over 10 runs goes below 5% for the LIL method.



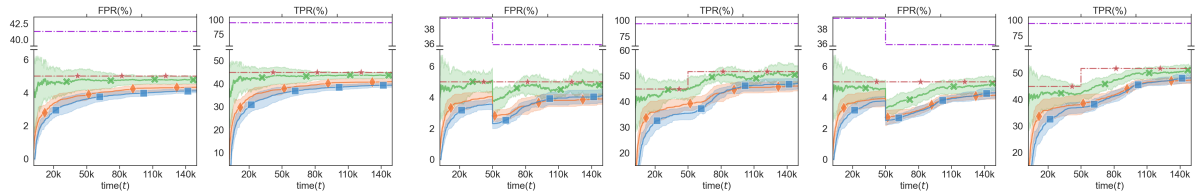
(a) No distribution shift, no window. (b) Distribution shift, 5k window. (c) Distribution shift, 10k window.

Figure E.4: Results with the MDS scores on CIFAR-10 as ID dataset. For (b) and (c) the distribution shifts at $t = 50k$. The arrow indicates the time at which the mean FPR + std. deviation over 10 runs goes below 5% for the LIL method.



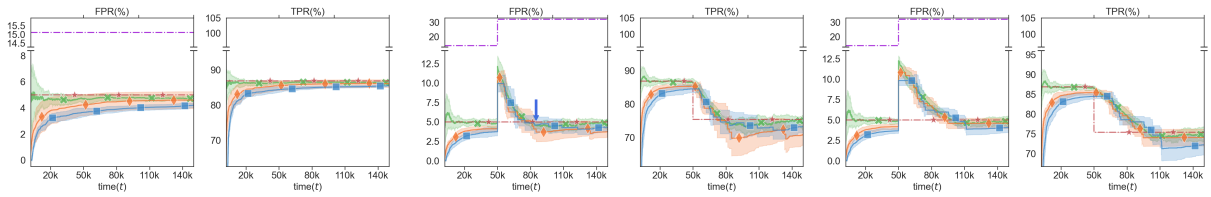
(a) No distribution shift, no window. (b) Distribution shift, 5k window. (c) Distribution shift, 10k window.

Figure E.5: Results with the VIM scores on CIFAR-10 as ID dataset. For (b) and (c) the distribution shifts at $t = 50k$. The arrow indicates the time at which the mean FPR + std. deviation over 10 runs goes below 5% for the LIL method.



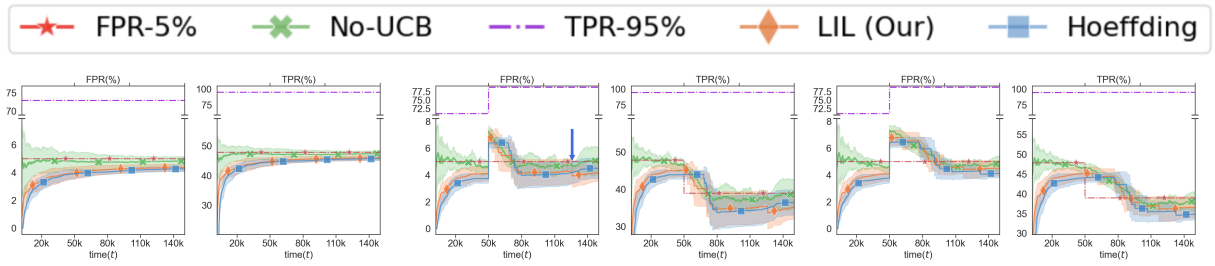
(a) No distribution shift, no window. (b) Distribution shift, 5k window. (c) Distribution shift, 10k window.

Figure E.6: Results with the ODIN scores on CIFAR-10 as ID dataset. For (b) and (c) the distribution shifts at $t = 50k$. The arrow indicates the time at which the mean FPR + std. deviation over 10 runs goes below 5% for the LIL method.



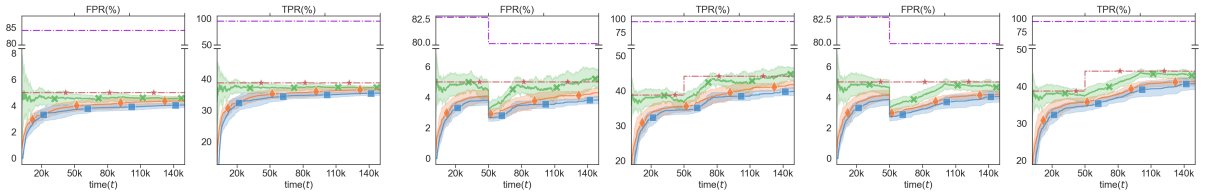
(a) No distribution shift, no window. (b) Distribution shift, 5k window. (c) Distribution shift, 10k window.

Figure E.7: Results with the SSD scores on CIFAR-10 as ID dataset. For (b) and (c) the distribution shifts at $t = 50k$. The arrow indicates the time at which the mean FPR + std. deviation over 10 runs goes below 5% for the LIL method.



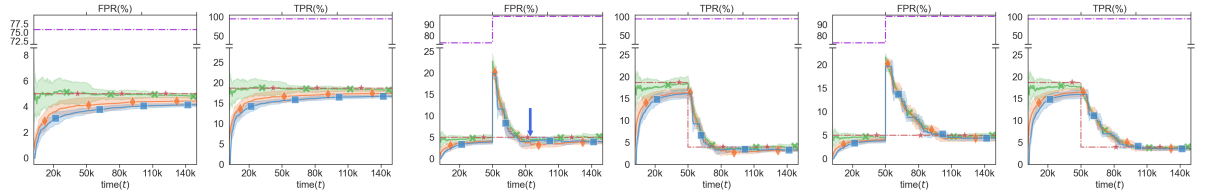
(a) No distribution shift, no window. (b) Distribution shift, 5k window. (c) Distribution shift, 10k window.

Figure E.8: Results with the KNN scores on CIFAR-100 as ID dataset. For (b) and (c) the distribution shifts at $t = 50k$. The arrow indicates the time at which the mean FPR + std. deviation over 10 runs goes below 5% for the LIL method.



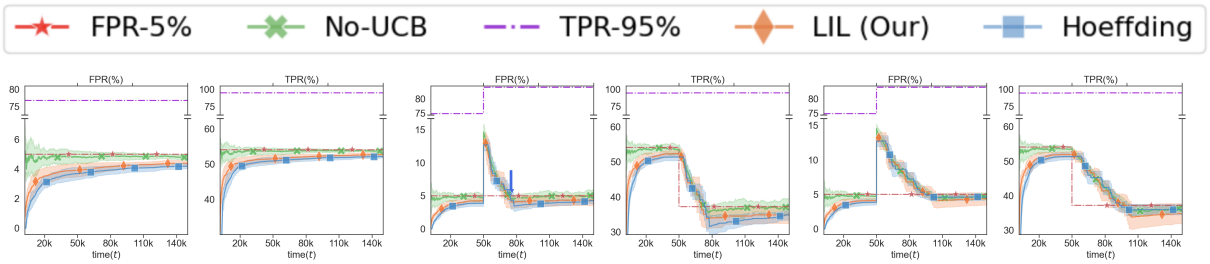
(a) No distribution shift, no window. (b) Distribution shift, 5k window. (c) Distribution shift, 10k window.

Figure E.9: Results with the EBO scores on CIFAR-100 as ID dataset. For (b) and (c) the distribution shifts at $t = 50k$. The arrow indicates the time at which the mean FPR + std. deviation over 10 runs goes below 5% for the LIL method.



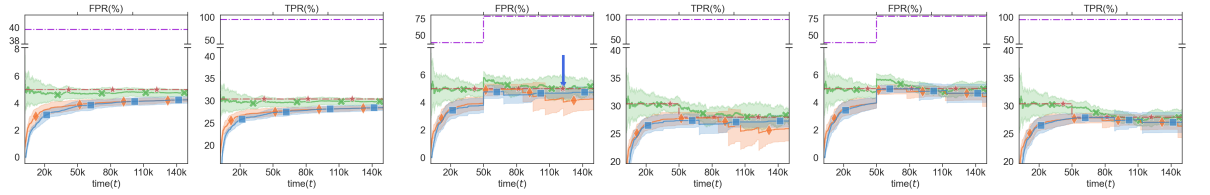
(a) No distribution shift, no window. (b) Distribution shift, 5k window. (c) Distribution shift, 10k window.

Figure E.10: Results with the MDS scores on CIFAR-100 as ID dataset. For (b) and (c) the distribution shifts at $t = 50k$. The arrow indicates the time at which the mean FPR + std. deviation over 10 runs goes below 5% for the LIL method.



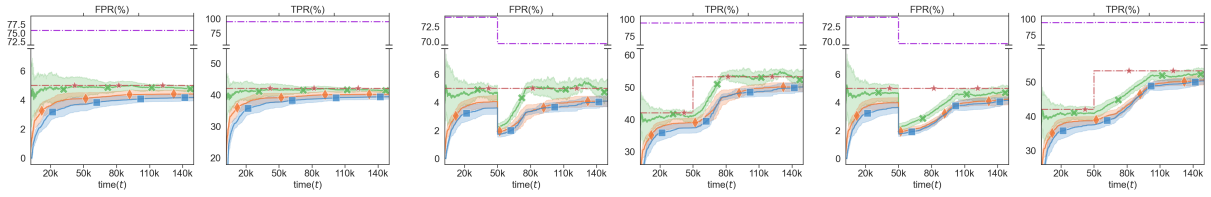
(a) No distribution shift, no window. (b) Distribution shift, 5k window. (c) Distribution shift, 10k window.

Figure E.11: Results with the VIM scores on CIFAR-100 as ID dataset. For (b) and (c) the distribution shifts at $t = 50k$. The arrow indicates the time at which the mean FPR + std. deviation over 10 runs goes below 5% for the LIL method.



(a) No distribution shift, no window. (b) Distribution shift, 5k window. (c) Distribution shift, 10k window.

Figure E.12: Results with the SSD scores on CIFAR-100 as ID dataset. For (b) and (c) the distribution shifts at $t = 50k$. The arrow indicates the time at which the mean FPR + std. deviation over 10 runs goes below 5% for the LIL method.



(a) No distribution shift, no window. (b) Distribution shift, 5k window. (c) Distribution shift, 10k window.

Figure E.13: Results with the ODIN scores on CIFAR-100 as ID dataset. For (b) and (c) the distribution shifts at $t = 50k$. The arrow indicates the time at which the mean FPR + std. deviation over 10 runs goes below 5% for the LIL method.

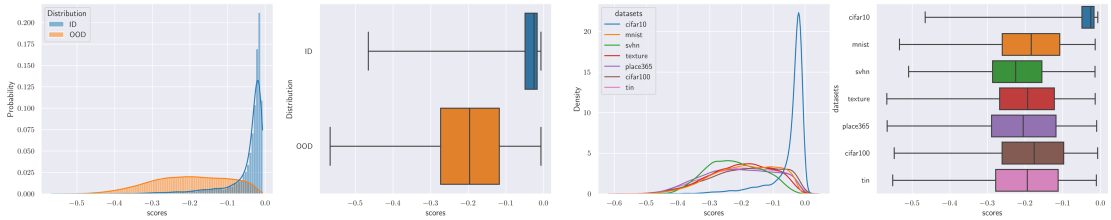


Figure E.14: Scores distribution for KNN with CIFAR-10 as In-Distribution.

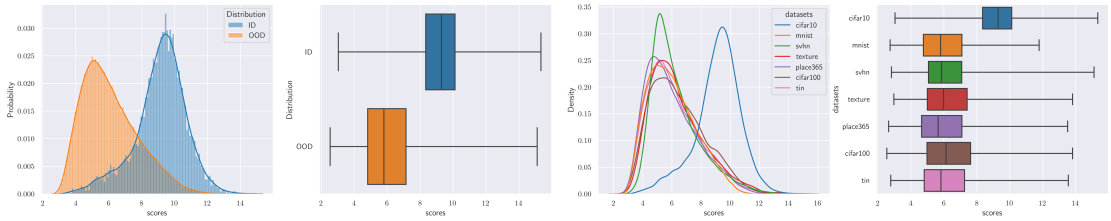


Figure E.15: Scores distribution for EBO with CIFAR-10 as In-Distribution.

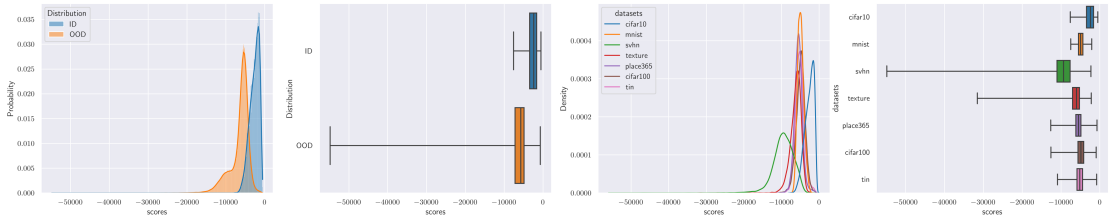


Figure E.16: Scores distribution for SSD with CIFAR-10 as In-Distribution.

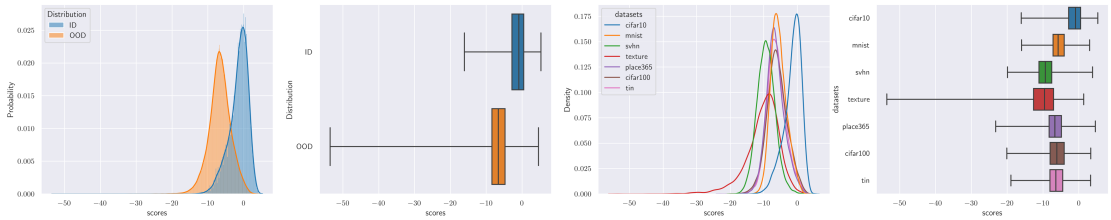


Figure E.17: Scores distribution for VIM with CIFAR-10 as In-Distribution.

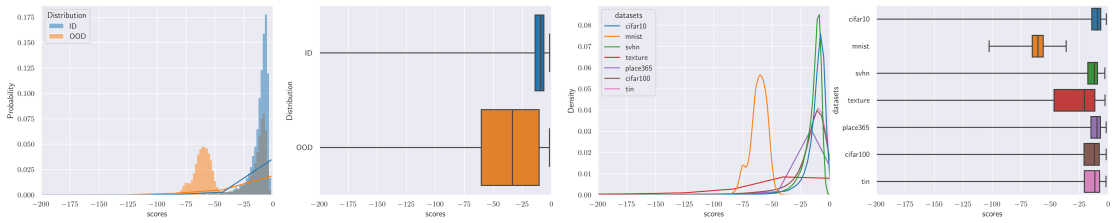


Figure E.18: Scores distribution for MDS with CIFAR-10 as In-Distribution.

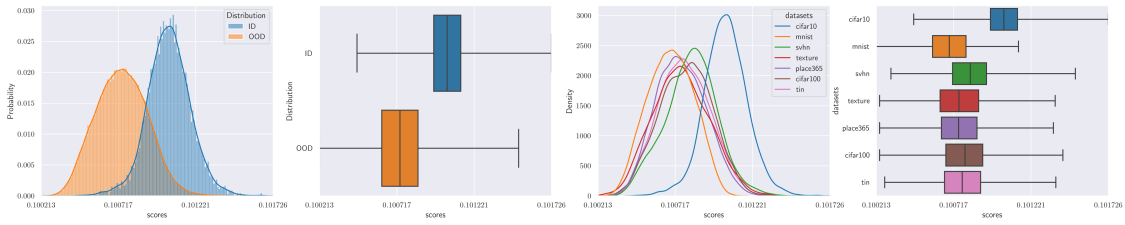


Figure E.19: Scores distribution for ODIN with CIFAR-10 as In-Distribution.

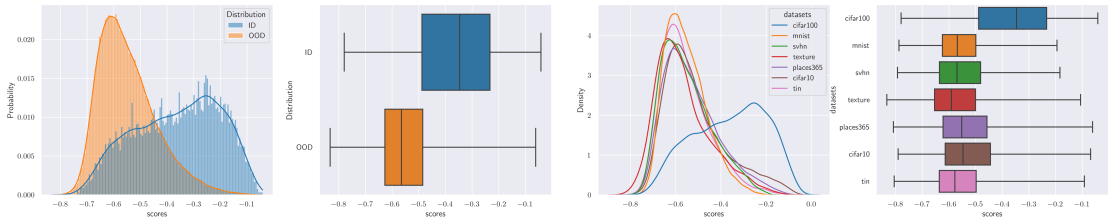


Figure E.20: Scores distribution for KNN with cifar-100 as In-Distribution.

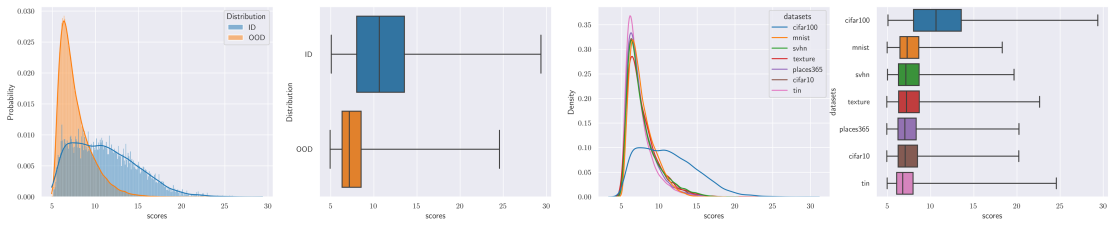


Figure E.21: Scores distribution for EBO with Cifar-100 as In-Distribution.

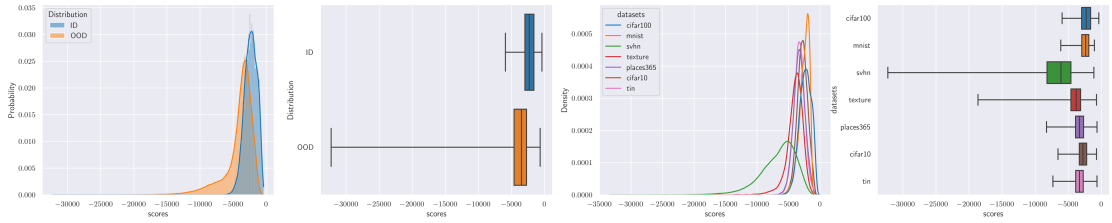


Figure E.22: Scores distribution for SSD with Cifar-100 as In-Distribution.

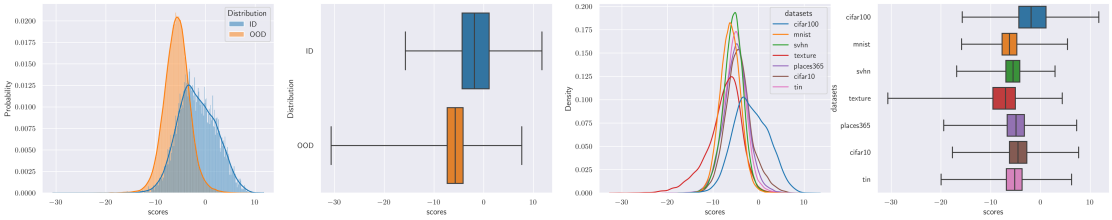


Figure E.23: Scores distribution for VIM with Cifar-100 as In-Distribution.

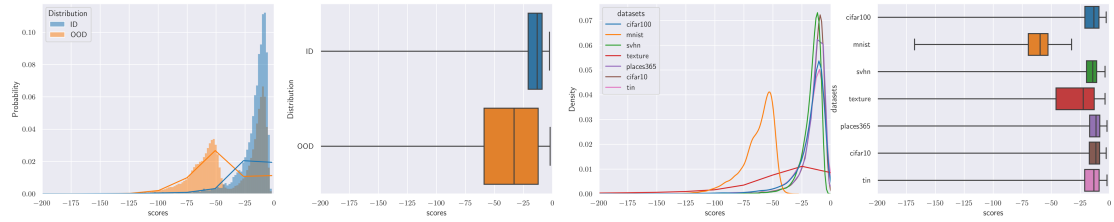


Figure E.24: Scores distribution for MDS with Cifar-100 as In-Distribution.

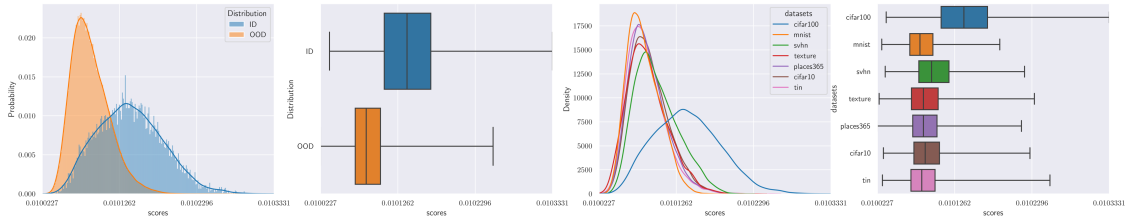


Figure E.25: Scores distribution for ODIN with Cifar-100 as In-Distribution.

Appendix F

Appendix to Chapter 9

The appendix is organized as follows. We provide a summary of the variables and constants in Table F.1. Then, we give an algorithm for the entire ASAT workflow in Section F.2 and detailed proofs for the FPR control in Section F.3. In Section F.4, we provide additional experimental results and details of the experiment protocol and hyperparameters used for the experiments.

F.1 Glossary

We provide the following list of variables and constants used in our framework (Table F.1).

F.2 Additional Details for Methodology

We provide a detailed description of the ASAT workflow in Algorithm 13. We also provide the algorithm for binary search procedures used to solve (Q1) in Algorithm 14.

DKW Criterion for Model Selection. Suppose an update occurs at time t (i.e., $\Delta_t^{ood} \geq \omega_{U_t}^{ood}$) by solving (P9.2). We ensure that the updated scoring function $\widehat{g}_t$ outperforms $\widehat{g}_{t-1}$, by applying following TPR criterion:

$$\widehat{\text{TPR}}(\widehat{g}_t, \widehat{\lambda}_t) + 2\zeta(\delta, |X_0^{(id)}|) > \widehat{\text{TPR}}(\widehat{g}_{t-1}, \widehat{\lambda}_{t-1}),$$

Algorithm 13: ASAT (Adaptive Scoring function, Adaptive Threshold)

procedure ASAT(α, p, δ, β, $\widehat{g}_0$, $\{\omega_i^{(ood)}\}_{i\geq 0}$, $X_0^{(id)}$)
  $\widehat{\lambda}_0 \leftarrow +\infty$, $X_0^{(ood)} \leftarrow \emptyset, U_0 \leftarrow 1$.
  $\Delta_0^{(ood)} \leftarrow 0$ (# new OOD identified via human labels since last update)
  for $t = 1, 2, \ldots$ **do**
    Observe x_t
    Compute $s_t \leftarrow \widehat{g}_{t-1}(x_t)$
    if $s_t \leq \widehat{\lambda}_{t-1}$ **then** ▷x_t is predicted as OOD
      $l_t \leftarrow 1$ ▷$l_t = 1$ w.p. 1
    else ▷x_t is predicted as ID
      $l_t \sim \text{Bern}(p)$ ▷$l_t = 1$ w.p. p
    end if
    if $l_t = 1$ **then**
      $y_t =$ GetHumanLabel(x_t) ▷Obtain human label y_t if $l_t = 1$
      if $y_t = 0$ **then** ▷If x_t is truly OOD
        $X_t^{(ood)} \leftarrow X_{t-1}^{(ood)} \cup x_t$ ▷Add x_t to the set of OOD
        $\Delta_t^{(ood)} \leftarrow \Delta_t^{(ood)} + 1$
      end if
    end if
    ▷ Learn new scoring function
    if $\Delta_t^{(ood)} \geq \omega_{U_t}^{(ood)}$ **then** ▷If we have collected enough new OOD samples
      $i \leftarrow i + 1, \quad \Delta_t^{(ood)} \leftarrow 0$
      $\widehat{g}_t, \widehat{\lambda}'_t \leftarrow \underset{g\in\mathcal{G},\lambda\in\Lambda}{\arg\min} -\widetilde{\text{TPR}}(g,\lambda) + \beta \cdot \widetilde{\text{FPR}}_t(g,\lambda)$ using $X_t^{(ood)}$ and $X_0^{(id)}$ (P9.2)
      ▷ Estimate threshold
      $\widehat{\lambda}_t \leftarrow \underset{\lambda\in\Lambda}{\arg\min}\, \lambda$ s.t. $\widehat{\text{FPR}}_t(\widehat{g}_t,\lambda) + \psi_t(\delta) \leq \alpha$ via BinarySearch($X_t^{(ood)}$, $\widehat{g}_t$) (Q1)
      ▷ Model selection
      if $\widehat{\text{TPR}}(\widehat{g}_t,\widehat{\lambda}_t) + 2\zeta(\delta, |X_0^{(id)}|) > \widehat{\text{TPR}}(\widehat{g}_{t-1},\widehat{\lambda}_{t-1})$ **then** ▷If the selection criterion is met.
        $\widehat{g}_t \leftarrow \widehat{g}_t$ ▷Deploy the new scoring function
        $\widehat{\lambda}_t \leftarrow \widehat{\lambda}_t$ ▷Update to the new threshold
        $U_t \leftarrow U_{t-1} + 1$
      else
        $\widehat{g}_t \leftarrow \widehat{g}_{t-1}$ ▷Retain the old scoring function
        $U_t \leftarrow U_{t-1}$
      end if
    end if
    if g is not updated **then**
      ▷ Estimate threshold
      $\widehat{\lambda}_t \leftarrow \underset{\lambda\in\Lambda}{\arg\min}\, \lambda$ s.t. $\widehat{\text{FPR}}_t(\widehat{g}_t,\lambda) + \psi_t(\delta) \leq \alpha$ via BinarySearch(X_t^{ood}, $\widehat{g}_t$) (Q1)
    end if
    if $l_t = 1$ **then**
      Output y_t ▷Output the true label obtained via human feedback
    else
      Output $\widehat{y}_t := \mathbb{1}\{s_t > \widehat{\lambda}_{t-1}\}$ ▷Output predicted label via one-sided threshold function
    end if
  end for
end procedure

Algorithm 14: Binary Search to Solve (Q1)

procedure BINARYSEARCH($X_t^{(ood)}$, $\widehat{g}_t$)
  $\lambda_l = \Lambda_{min}$, $\lambda_h = \Lambda_{max}$, $feasible =$ `False`, $\epsilon \approx 0$
  while $\lambda_h - \lambda_l > \epsilon$ **do**
    $\lambda_m = (\lambda_l + \lambda_h)/2$
    if $\widehat{\mathrm{FPR}}_t(\widehat{g}_t, \lambda_m) + \psi_t(\delta) \leq \alpha$ **then**
      $feasible =$ `True`
      $\lambda_h = \lambda_m$
    else
      $\lambda_l = \lambda_m$
    end if
  end while
  return λ_m, $feasible$
end procedure

where $\zeta(\delta, |X_0^{(id)}|)$ is the Dvoretzky–Kiefer–Wolfowitz (DKW) confidence interval (Nadaraya, 1965) given by

$$\zeta(\delta, |X_0^{(id)}|) = \sqrt{\frac{1}{|X_0^{(id)}|} \log\left(\frac{2}{\delta}\right)},$$

which is valid w.p. at least $1 - \delta$. This confidence bound relies on the fact that $X_0^{(id)}$ is i.i.d. and the TPR can be expressed as $\mathrm{TPR}(g, \lambda) = 1 - F_1(\lambda; g)$, where F_1 is the CDF of the ID score distribution mapped by g.

F.3 Proofs

Summary of ASAT Settings. At each time t, the ASAT system processes an input x_t using scoring function $\widehat{g}_{t-1}$ and threshold $\widehat{\lambda}_{t-1}$, both learned at the previous time $t-1$. It first computes score $s_t = \widehat{g}_{t-1}(x_t)$ for x_t. If $s_t < \widehat{\lambda}_{t-1}$, then x_t is considered OOD and receives a human label $y_t \in \{0, 1\}$. If $s_t \geq \widehat{\lambda}_{t-1}$, x_t is considered ID, and in this case, we obtain the human label y_t only with probability $p \in (0, 1)$ (i.e., importance sampling). Now, if x_t is identified as OOD and the system has collected enough new OOD points required for the next update (i.e., $\Delta_{U_t}^{(ood)}$), we estimate the FPR and TPR (9.1) using the collected finite samples to solve the optimization in (P9.2).

The FPR and TPR estimates are defined as

$$\widehat{\mathrm{FPR}}_t(g,\lambda) := \frac{\sum_{u=1}^{t} Z_u \cdot \mathbb{1}\{g(x_u) > \lambda\}}{\sum_{u=1}^{t} Z_u}, \tag{F.1}$$

$$Z_u := \begin{cases} 1 & \text{if } s_u < \widehat{\lambda}_{u-1}, i_u = 0, y_u = 0, \\ \frac{1}{p} & \text{if } s_u \geq \widehat{\lambda}_{u-1}, i_u = 1, y_u = 0, \\ 0 & \text{otherwise,} \end{cases}$$

and

$$\widehat{\mathrm{TPR}}(g,\lambda) := \frac{1}{|X_0^{(id)}|} \sum_{x \in X_0^{(id)}} \mathbb{1}\{g(x) \geq \lambda\}. \tag{F.2}$$

In (P9.2), the differentiable surrogates of the FPR and TPR estimates from (9.4) and (9.5) are used for efficient optimization. If an update occurs, the system obtains a new scoring function $\widehat{g}_t$ and threshold $\widehat{\lambda}'_t$. Otherwise, the previous scoring function is retained, i.e., $\widehat{g}_t = \widehat{g}_{t-1}$. Since the optimization problem in (P9.2) is unconstrained, there's no guarantee that the original FPR constraint will be satisfied by $\widehat{g}_t$ and $\widehat{\lambda}'_t$. Therefore, the system solves a different optimization in (Q1) to find $\widehat{\lambda}_t$, replacing $\widehat{\lambda}'_t$, based on the current scoring function $\widehat{g}_t$. In this optimization, we use the FPR estimate in eq (9.2) and LIL-based upper confidence bounds (UCB) to ensure that the true FPR constraint is satisfied with high probability. Additional details on the workflow are provided in F.2.

Proof Outline. Our goal in ASAT is to ensure that $\mathrm{FPR}(\widehat{g}_t, \widehat{\lambda}_t)$ remains below α at all times. Similar results were achieved for FSAT, where thresholds were adapted for a *fixed* scoring function (Vishwakarma et al., 2024c). Their results relied on constructing time-uniform FPR estimation bounds for all times t and thresholds $\lambda \in \Lambda$. The OOD samples used to estimate the FPR are non-i.i.d., as the decision to receive a human label for x_t depends on the previously learned $\widehat{g}_{t-1}$ and $\widehat{\lambda}_{t-1}$, which themselves are functions of past data $\{x_1, \ldots, x_{t-1}\}$. In FSAT, this dependency is handled by applying the martingale version of the LIL bound (Khinchine, 1924; Balsubramani, 2015), which leads to a time-uniform confidence sequence.

However, in ASAT, the same LIL-based UCB does not directly apply, as the system periodically adapts new, *time-varying*, scoring functions and uses them for predictions and sample collection. Despite this, we claim that our design still ensures comparable FPR control. We first show that each x_t identified as OOD is an unbiased estimator for the FPR.

Lemma F.3.1. *Let Z_u be defined as in eq (9.2), and let i_u be the indicator for whether x_u is importance sampled. Given $g \in \mathcal{G}$ and $\lambda \in \Lambda$, we have*

$$\mathbb{E}_{x_u,i_u}[Z_u \mathbb{1}\{g(x_u) \geq \lambda)\}] = FPR(g,\lambda).$$

Proof. Let $m_u = \mathbb{P}\left(\widehat{g}_{u-1}(x_u) < \widehat{\lambda}_{u-1}\right)$.

$$\mathbb{E}_{x_u,i_u}\left[Z_u \mathbb{1}\{g(x_u) \geq \lambda)\}\right] = \mathbb{E}_{x_u}\left[\mathbb{1}\{g(x_u) \geq \lambda)\}\mathbb{E}_{i_u|x_u}\left[Z_u\right]\right].$$

Conditioned on x_u, we have

$$\begin{aligned}
\mathbb{E}_{i_u|x_u}\left[Z_u\right] &= \mathbb{P}(\widehat{g}_{u-1}(x_u) < \widehat{\lambda}_{u-1} \mid x_u) \cdot \mathbb{P}(i_u = 0) + \frac{1}{p}\mathbb{P}(\widehat{g}_{u-1}(x_u) < \widehat{\lambda}_{u-1} \mid x_u) \cdot \mathbb{P}(i_u = 1) \\
&= m_u + \frac{(1-m_u)p}{p} \\
&= 1.
\end{aligned}$$

Hence, we have

$$\mathbb{E}_{x_u,i_u}\left[Z_u \mathbb{1}\{g(x_u) \geq \lambda)\}\right] = \mathbb{E}_{x_u}\left[\mathbb{1}\{g(x_u) \geq \lambda)\}\right] = \mathrm{FPR}(g,\lambda).$$

□

Next, we identify a martingale structure within the OOD samples. This step is crucial, as it will allow us to apply the martingale-LIL, yielding a time-uniform confidence sequence.

Lemma F.3.2. *For any $g \in \mathcal{G}$, $\lambda \in \Lambda$ and $t \geq 1$, the stochastic process $\{M_t(g,\lambda)\}_{t\geq 1}$ given by*

$$M_t(g,\lambda) := \sum_{u=1}^{t} \Big(Z_u \mathbb{1}\{g(x_u) > \lambda\} - FPR(g,\lambda) \Big)$$

is a martingale with respect to the filtration $\{\mathcal{F}_u\}_{u=1}^{t-1}$, where $\mathcal{F}_u$ is the σ-field generated by the events until time u.

Proof. We observe that $\mathbb{E}M_t < \infty$ and M_t is $\mathcal{F}_t$-measurable. Now, we have

$$\begin{aligned}
\mathbb{E}[M_t(g,\lambda) \mid \mathcal{F}_{t-1}] &= \mathbb{E}\Big[Z_t \mathbb{1}\{g(x_t) > \lambda\} - \mathrm{FPR}(g,\lambda) + M_{t-1}(g,\lambda) \mid M_{t-1}(g,\lambda) \Big] \\
&= \mathbb{E}\Big[Z_t \mathbb{1}\{g(x_t) > \lambda\} - \mathrm{FPR}(g,\lambda)] + M_{t-1}(g,\lambda) \Big] \\
&= M_{t-1}(g,\lambda),
\end{aligned}$$

where the third equality follows from Lemma (F.3.1). □

Lemma F.3.2 leads to the following proposition on the LIL-based time-uniform confidence intervals for $\{M_t(g,\lambda)\}$.

Proposition 4. Fix $g \in \mathcal{G}$. For any $t \geq 1$, let $N_t^{(o)} = \sum_{u=1}^{t} Z_u$ and $N_t^{(imp)}$ be the number of importance sampled OOD points and $\beta_t = \frac{N_t^{imp}}{N_t^{(o)}}$. Let $c_t := 1 + \frac{1-p}{p^2}\beta_t$, and $t_0 = \min\{u : c_u N_u^{(o)} \geq 173 \log(4/\delta)\}$. M_t is defined as in Lemma F.3.2. Let $\Lambda := \{\Lambda_{\min}, \Lambda_{\min} + \eta, ..., \Lambda_{\max}\}$, with discretization parameter $\eta > 0$. Then, we have

$$\mathbb{P}\left(\forall\, t \geq t_0 : \sup_{\lambda \in \Lambda} |M_t(g,\lambda)| < \psi'(t,\delta) \right) \geq 1 - \delta,$$

where

$$\psi_t'(\delta) := \sqrt{3 c_t N_t^{(o)} \left[2 \log\log\left(\frac{3 c_t N_t^{(o)}}{2} \right) + \log\left(\frac{2|\Lambda|}{\delta} \right) \right]}.$$

Proof. For the martingale sequence $\{M_t(g,\lambda)\}_{t\geq 1}$ defined in Lemma (F.3.2), the following is true

for all $t \geq 1$,

$$|M_t - M_{t-1}| \leq \begin{cases} 1 & : s_u < \widehat{\lambda}_{u-1}, i_u = 0, y_u = 0 \\ 1/p & : s_u < \widehat{\lambda}_{u-1}, i_u = 0, y_u = 0 \\ 0 & : o.w. \end{cases}$$

Hence, we know that $\{M_t\}$ has a bounded increment for all t. Define now $N_t^{(o)} = \sum_{u=1}^{t} Z_u$ and $N_t^{(imp)}$ be the number of importance sampled OOD points. Set $\beta_t = \frac{N_t^{imp}}{N_t^{(o)}}$, $c_t := 1 + \frac{1-p}{p^2}\beta_t$, and $t_0 = \min\{u : c_u N_u^{(o)} \geq 173 \log(4/\delta)\}$. Then, applying the Martingale LIL result from Balsubramani (2015), we obtained the following uniform concentration for a given $g \in \mathcal{G}$ and $\lambda \in \Lambda$:

$$\mathbb{P}\left(\forall\, t \geq t_0 : |M_t(g,\lambda)| \leq \sqrt{3c_t N_t^{(o)} \left[2 \log\log\left(\frac{3c_t N_t^{(o)}}{2}\right) + \log\left(\frac{2|\Lambda|}{\delta}\right)\right]}\right) \geq 1 - \delta.$$

The result follows by applying the union bound over the discretized class Λ of λ. □

Proposition 4 provides a time-uniform confidence sequence valid for all $t \geq t_0$ and $\lambda \in \Lambda$, but it holds for a fixed $g \in \mathcal{G}$ and must be adapted to account for *time-varying* scoring functions g_t for all $t \geq t_0$. The following lemma gives the desired result.

Proposition 5. Let U_t be the number of updates before time t. Let $N_t^{(o)} = \sum_{u=1}^{t} Z_u$, and $N_t^{(imp)}$ be the number of importance sampled OOD points. Define $\beta_t = N_t^{(imp)}/N_t^{(o)}$, $c_t = 1+(1-p)\beta_t/p^2$, and $t_0 = \min\{u : c_u N_u^{(o)} \geq 173 \log(4/\delta)\}$. Let $\Lambda := \{\Lambda_{\min}, \Lambda_{\min}+\eta, \ldots, \Lambda_{\max}\}$, with discretization parameter $\eta > 0$. Then, we have

$$\mathbb{P}\left(\forall\, t \geq t_0 : \sup_{\lambda \in \Lambda} |\widehat{\mathrm{FPR}}_t(g_t,\lambda) - \mathrm{FPR}(g_t,\lambda)| < \psi_t(\delta)\right) \geq 1 - \delta,$$

where

$$\psi_t(\delta) := \sqrt{\frac{3c_t}{N_t^{(o)}}\left[2 \log\log\left(\frac{3c_t N_t^{(o)}}{2}\right) + 2\log\left(\frac{4U_t|\Lambda|}{\delta}\right)\right]}.$$

Proof. The result follows from Proposition 4 by scaling with $N_t^{(o)}$ and applying the union bound

over the set of U_t different scoring functions updated until time t. The union bound is done as follows,

Let the failure probability δ'_t denote the probability that the bound is invalid at time t. Consider, $\delta'_t = \frac{\delta}{2U_t^2}$. Thus, the overall failure probability $= \sum_{U_t=1}^{\infty} \frac{\delta}{2U_t^2} = \delta\frac{\pi^2}{12} < \delta$.

Each time the scoring function is updated, we invoke the bound in Proposition 4 with failure probability $\frac{\delta}{2U_t^2}$. This gives the desired bound.

□

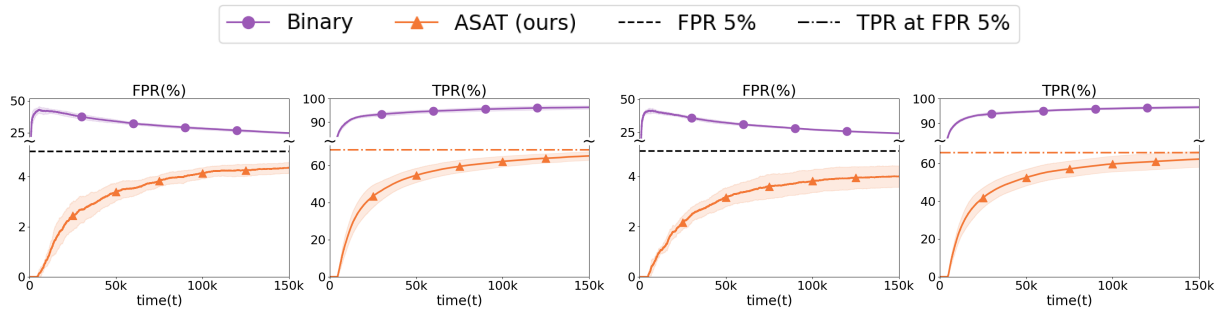
Proposition 5 establishes the desired safety guarantees for FPR control, ensuring that with probability at least $1-\delta$, $\text{FPR}(\widehat{g}_t, \widehat{\lambda}_t) \leq \alpha$ for all $t \geq t_0$ in the ASAT system. This comes with the trade-off, as $\psi_t(\delta)$ now contains the additional term U_t—the number of total updates—in the second logarithm term. In practice, however, U_t is expected to be much smaller than $N_t^{(o)}$, as updates only occur periodically after collecting enough new OOD samples. As a result, the additional dependency on U_t introduces only a minor increase in $\psi_t(\delta)$ compared to FSAT. This slight increase is outweighed by the advantage of using the learned $\widehat{g}_t$, which optimizes TPR.

F.4 Additional Experiments and Details

F.4.1 Additional Baseline

For completeness, we compare ASAT against a naive binary classifier baseline. This baseline uses a two-layer neural network (with two output neurons) that matches the architecture and parameter size of ASAT, and is trained with cross-entropy loss on the ID and OOD data collected up to each time t. Since this classifier is not scoring-based, we do not use importance sampling. We update the classifier with the same frequency as ASAT, using the AdamW optimizer. Initially, the classifier is randomly initialized and predicts all samples as OOD until enough data is collected for its first update.

As shown in Figure F.1, while the binary classifier can distinguish between ID and OOD points, it incurs significant FPR violations (exceeding 5%) because it does not explicitly optimize to



(a) Stationary, KNN, CIFAR-100 as OOD. (b) Stationary, KNN, Near-OOD as OOD.

Figure F.1: Comparison between ASAT and a naive binary classifier baseline. ASAT is initialized with KNN. CIFAR-10 as ID and CIFAR-100 and Near-OOD as OOD. Each method is repeated five times and is shown with mean and std. deviation. The binary classifier is trained using cross-entropy loss on the ID and OOD data collected up to time t, without importance sampling (since the classification is not based on score thresholding).

maintain a low FPR while maximizing TPR. In contrast, ASAT adapts both thresholds and scoring functions to maximize TPR while ensuring strict FPR control. Hence, these results confirm that our ASAT framework successfully meets its dual objectives, offering a more reliable solution (via FPR control) for OOD detection.

F.4.2 Searching for Constants in Heuristic LIL UCB

We derived the following theoretical LIL-based upper confidence bound (UCB) to ensure FPR control for ASAT with time-varying scoring functions $\widehat{g}_t$ for $t \geq 1$:

$$\psi_t(\delta) := \sqrt{\frac{3c_t}{N_t^{(o)}}\left[2\log\log\left(\frac{3c_t N_t^{(o)}}{2}\right) + \log\left(\frac{2U_t|\Lambda|}{\delta}\right)\right]}, \tag{F.3}$$

where U_t is the total number of scoring functions deployed in the system by time t. The inclusion of U_t reflects the union bound required for FPR control given the time-varying scoring functions (see Appendix F.3). A similar UCB appears in the FSAT framework (Vishwakarma et al., 2024a), but without the U_t term and with different definitions of c_t and $N_t^{(o)}$. However, due to pessimistic constants in eq. (9.6) (e.g., $|\Lambda|$ may be large for small η), we define a heuristic UCB:

$$\psi_t^H(\delta) := c_1\sqrt{\frac{c_t}{N_t^{(o)}}\left[\log\log\left(c_2 c_t N_t^{(o)}\right) + \log\left(\frac{c_3}{\delta}\right)\right]},$$

with constants c_1, c_2, c_3. In FSAT, $c_1 = 0.5$, $c_2 = 0.75$, and $c_3 = 1.0$ were chosen via simulation of a p-baised coin to keep failure probability under 5%. In the ASAT main experiments, where we fix the optimization frequencies, we retain c_2 and c_3 but increase c_1 to 0.65 to account for the effect of U_t. The choice of c_1 depends on the number of scoring function updates, which requires adjustment if the optimization frequency changes.

F.4.3 Choice of $\mathcal{G}$

Our ASAT framework is flexible in the choices of $\mathcal{G}$. In the main experiments, we use a class $\mathcal{G}_1$ of two-layer neural networks defined on top of the ResNet18 model (He et al., 2015). Let $z := r(x)$ be the output feature from the penultimate layer of the Resnet18, denoted as r, for an image x. A network $g \in \mathcal{G}_1$ takes z as input and outputs a score $s \in \mathcal{S}$:

$$\mathcal{G}_1 := \{g : g(x) := \boldsymbol{W_2}\,\mathrm{ReLU}\,(\boldsymbol{W_1} r(x))\}\,,$$

where $\boldsymbol{W_1}, \boldsymbol{W_2}$ are learnable weight matrices of the two layers. The optimization procedures are detailed in the next section. This class of scoring functions leverages pre-extracted features for ID/OOD points and adapts the scoring functions *from scratch* to the specific OOD instances encountered by the system. This design is motivated by the observation that the backbone already performs heavy representation learning, and the 2-layer neural network as the scoring function only needs to learn a separation between ID and OOD samples, which makes the training computationally lightweight and scalable.

F.4.4 Searching for Hyper-parameters

We determine the hyperparameters and their values for the optimization problem in (P9.2) over $\mathcal{G}_1$ via grid search. First, we find the best-performing hyperparameter combination for training $g \in \mathcal{G}_1$ in *offline* settings using i.i.d. OOD and ID datasets. Then, we extend these choices to the ASAT settings, where only *dependent* OOD datasets are available. We verify that the same hyperparameters perform well in *online* settings, made possible by importance sampling techniques that preserve accurate FPR estimation in (9.2).

Table F.2 lists the hyperparameters and their values we explored. Specifically, we run experiments on every different combination of these lists of hyperparameters. We use the AdamW optimizer (Kingma and Ba, 2017), with learning rates of 0.0001 for g and 0.01 for λ, and a weight decay rate of 0.001 for L1 regularization for both g and λ. We also set $\beta = 1.5$ and $\kappa = 50$.

F.4.5 Effects of Different Updating Frequencies

We define the optimization frequencies $\omega_i^{(ood)}$ as the number of newly human-labeled OOD points required since the last update for the $(i+1)$th update of the scoring functions. In stationary settings, we fix $\omega_i^{(ood)}$ to be 100 for $i \in \{1, ..., 20\}$, 500 for $i \in \{21, 40\}$, and 1000 thereafter for consistency. This approach enables more frequent updates early on, with the frequency gradually decreasing as scoring functions improve. In nonstationary settings, we set $\omega_i^{(ood)}$ uniformly to avoid excessively infrequent updates after a shift. Additionally, $\omega_i^{(ood)}$ influences the width of LIL-based UCB. More frequent updates result in larger U_t, leading to a wider UCB. Thus, $\omega_i^{(ood)}$ presents a trade-off between faster OOD adaptation and larger UCB. More frequent updates of the scoring functions could also impact time efficiency, as each update requires additional computational resources.

To illustrate this, we run additional experiments in stationary settings with four different choices for $\omega_i^{(ood)}$: 100, 500, 1000, and 2500 uniformly for all $i \geq 1$. We fix $c_1 = 0.50$ for $\psi_t^H(\delta)$ across all frequencies, knowing this value is not large enough to provide strict FPR guarantees. This ensures a fair comparison across different frequencies. The results, shown in Figure (F.2), confirm our hypotheses: more frequent updates lead to faster adaptations but also result in a larger UCB width. More frequent updates impact time efficiency, with rough averages from five runs showing increased time costs. However, they also lead to better TPR convergence, which requires further exploration. Further research on update frequencies is left for future work.

F.4.6 Additional Experiments with Different Scoring Functions

In the main experiments, we present results for EBO and KNN. Additionally, we run additional experiments using the following alternative scoring functions, with the same settings and constants

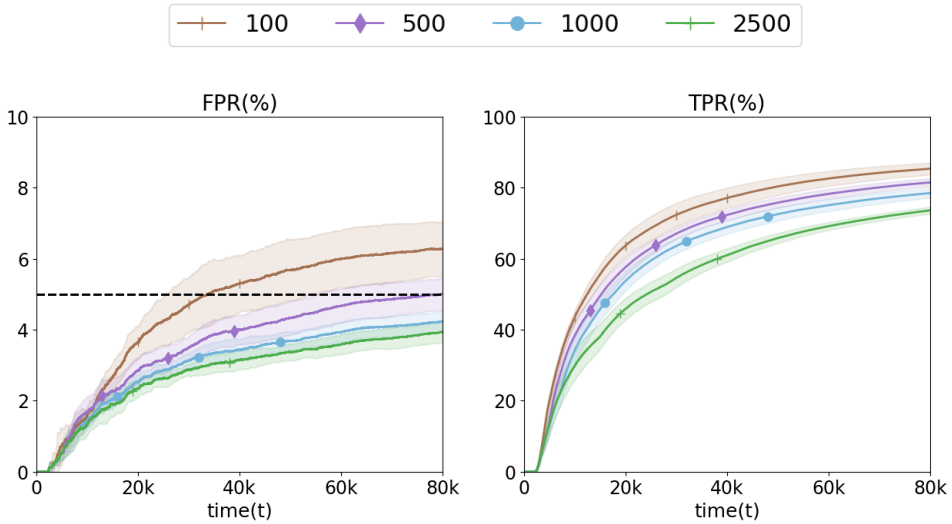


Figure F.2: Results on the stationary setting with different update frequencies, with each method repeated 5 times (mean and std shown). CIFAR-10 as ID and Places365 as OOD. Optimization frequencies are set uniformly for 100, 500, 1000, and 2500 new OOD samples. We fix $c_1 = 0.5$ in the heuristic UCB $\psi_t^H(\delta)$.

as in the main experiments. Note that while these scoring functions remain fixed in both FSFT and FSAT, the scoring function is immediately updated to a new scoring function from $\mathcal{G}_1$ in ASAT. We use these scoring functions adapted by Yang et al. (2022a).

1. **Energy Score (EBO).** EBO (Liu et al., 2020) uses an energy function derived from a discriminative model to distinguish OOD inputs from ID data. Energy scores are shown to align more with probability density and to be less prone to overconfidence than softmax scores.

2. **K-Nearest Neighbors (KNN).** KNN (Sun et al., 2022) uses non-parametric nearest-neighbors distance without strong assumptions about the feature space. In particular, KNN measures the distance between an input and its k closest ID neighbors. We use $k = 50$ as in Open-OOD benchmarks.

3. **Mahalanobis Distance (MDS).** MDS (Lee et al., 2018a) computes the Mahalanobis distance between an input and the nearest class-conditional Gaussian distribution and outputs a confidence score based on Gaussian discriminant analysis (GDA).

4. **Virtual Logits Matching (VIM).** VIM (Wang et al., 2022c) combines a class-agnostic score from the feature space with ID-dependent logits. It uses the residual of the feature against the principal space (we use dimension 10) and matches the logits with a constant scaling.

5. **OOD in Neural Networks (ODIN).** ODIN (Liang et al., 2017) uses softmax scores from DNNs, scales them with temperature, and applies gradient-based input perturbations to compute the score. We use a temperature of 1000 and a noise of 0.0014.

6. **Activation Shaping (ASH).** ASH (Djurisic et al., 2023) applies activation shaping, removing a large portion (e.g., 90%) of a sample's activation at a late layer, simplifying the rest. The method requires no extra training or significant model modifications.

7. **Simplified Hopfield Energy (SHE).** SHE (Zhang et al., 2023) computes a score based on Hopfield energy in a store-and-compare paradigm. It transforms penultimate layer features into stored patterns representing ID data, which serve as anchors for scoring new inputs.

Results for stationary and nonstationary OOD settings are shown in Figures F.3 and F.4, respectively.

F.4.7 Additional Experiments on Different ID Datasets

In the main experiments, we use CIFAR-10 as the ID dataset across all experiments. Additionally, we present results for CIFAR-100 (Deng et al., 2009) as the ID dataset. For stationary settings, we consider two OOD datasets: (i) CIFAR-10 and (ii) Places365. In the nonstationary OOD settings, we evaluate the scenario where a shift occurs at $t = 50$k, transitioning from a Far-OOD mixture (MNIST, SVHN, and Texture) to a Near-OOD mixture (CIFAR-10, Tiny-ImageNet, and Places365). Results for stationary and distribution-shift settings are shown in Figures F.5 and F.6, respectively.

In addition, we have experiments in a large-scale setting using ViT-B/16 (Dosovitskiy et al., 2021) as the feature extractor and ImageNet-1k (Deng et al., 2009) as ID. In the stationary setting, we used OpenImage-O as OOD and compared ASAT against FSAT and FSFT with MSP and KNN

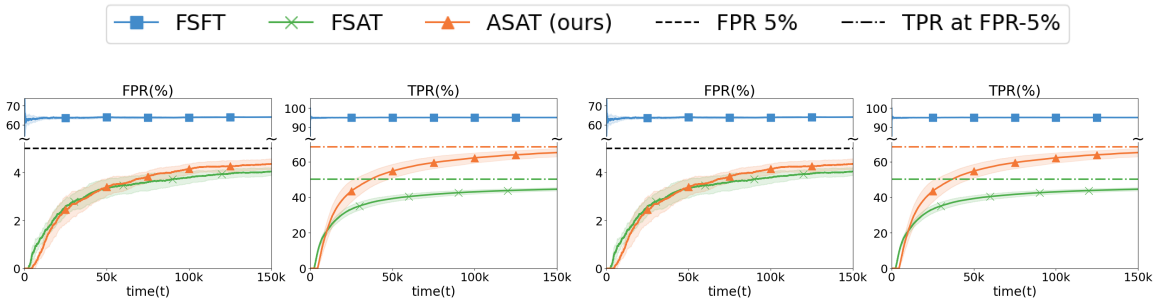
scoring functions. As expected, ASAT outperformed FSAT in TPR while maintaining strict FPR control, which confirms that ASAT scales well to large datasets like ImageNet-1k and modern architectures like ViT. Results are shown in Figure F.7.

Symbol	Definition/Description
$\mathcal{X}$	feature space.
$\mathcal{Y}$	label space $\{0,1\}$, 0 for OOD and 1 for ID.
$\mathcal{D}_0, \mathcal{D}_1$	the underlying distributions of OOD and ID points
γ	mixture ratio of OOD and ID distributions.
$x_t, s_t, y_t, \widehat{y}_t$	sample, score, true label, and predicted label at time t.
$X_t^{(ood)}$.	*dependent* OOD samples collected up to time t.
$X_0^{(id)}$	i.i.d. ID samples available before deployment.
$\mathrm{FPR}(g, \lambda)$	population level false positive rate defined by scoring function g and threshold λ.
$\mathrm{TPR}(g, \lambda)$	population level true positive rate defined by scoring function g and threshold λ.
$\widehat{\mathrm{FPR}}_t(g, \lambda)$	estimated FPR at time t, adjusted to account for importance sampling (see eq. (9.2)).
$\widehat{\mathrm{TPR}}(g, \lambda)$	estimated TPR fixed for all t, based on i.i.d. ID points (see eq. (9.3)).
$\widetilde{\mathrm{FPR}}_t(g, \lambda)$	smoothed $\widehat{\mathrm{FPR}}_t(g, \lambda)$ (see eq. (9.4)) used for optimization in eq. (P9.2)
$\widetilde{\mathrm{TPR}}(g, \lambda)$	smoothed $\widehat{\mathrm{TPR}}(g, \lambda)$ (see eq. (9.5)) used for optimization in eq. (P9.2).
$\mathrm{FPR}_t^{(eval)}$	evaluation FPR for experiments in eq. (9.7).
$\mathrm{TPR}_t^{(eval)}$	evaluation TPR for experiments in eq. (9.7).
$\mathcal{G}, \Lambda$	hypothesis class of scoring functions over $\mathcal{X}$ and space of thresholds.
(g^*, λ^*)	the optimal solution (non-unique) to the joint optimization problem in eq. (P9.1).
$(\widehat{g}_t, \widehat{\lambda}_t)$	scoring function and threshold learned at time t.
$\widehat{\lambda}'_t$	*unsafe* threshold obtained from eq. (P9.1) together with $\widehat{g}_t$.
i_u	indicator of whether x_t was importance sampled.
$\sigma(\kappa, z)$	sigmoid function for input z.
$\omega_i^{(ood)}$	the number of new OOD points required for ith updating.
β_t, c_t	time-dependent variables used for LIL-based confidence bounds (see Proposition 1).
$\Delta_t^{(ood)}$	number of *new* OOD points identified till time t since *last update*.
$N_t^{(imp)}$	number of OOD points identified via importance sampling up to time t.
U_t	number of total deployments of new scoring functions by time t.
$N_w^{(eval)}$	window size that system uses for computing evaluation FPR and TPR.
$N_w^{(est)}$	window size for estimation and optimization for nonstationary OOD settings.
p	importance sampling probability.
δ	failure probability for confidence intervals.
α	the threshold for FPR for all time points.
η	discretization parameter for Λ.
β	optimization parameter that controls a trade-off between TPR and FPR.
κ	the smoothness parameter of sigmoid functions
(c_1, c_2, c_3)	constants use for heuristic LIL bounds.
$\psi_t(\delta)$	LIL-based confidence interval at time t.
$\psi_t^H(\delta)$	LIL-based heuristic interval at time t.
$\zeta(\delta, n)$	DKW confidence interval for TPR.

Table F.1: Glossary of variables and symbols used in the framework.

Parameter	Values
optimizer	AdamW, SGD
learning rate for g	0.00001, 0.0001, 0.001, 0.01
learning rate for λ	0.0001, 0.001, 0.01, 0.1
weight decay rate for g	0.0001, 0.001, 0.01, 0.1, 0.0
weight decay rate for λ	0.0001, 0.001, 0.01, 0.1, 0.0
β in (P9.2)	0.5, 0.75, 1.0, 1.25, 1.5, 1.75, 2.0
κ for sigmoid smoothness	1, 10, 50, 100

Table F.2: Lists of hyper-parameters we swept over to find the best combination for training $g \in \mathcal{G}_1$ and $\lambda \in \Lambda$ in optimization (P9.2).



(a) MDS, CIFAR-10 as ID and CIFAR-100 as OOD. (b) MDS, CIFAR-10 as ID and Near-OOD as OOD.

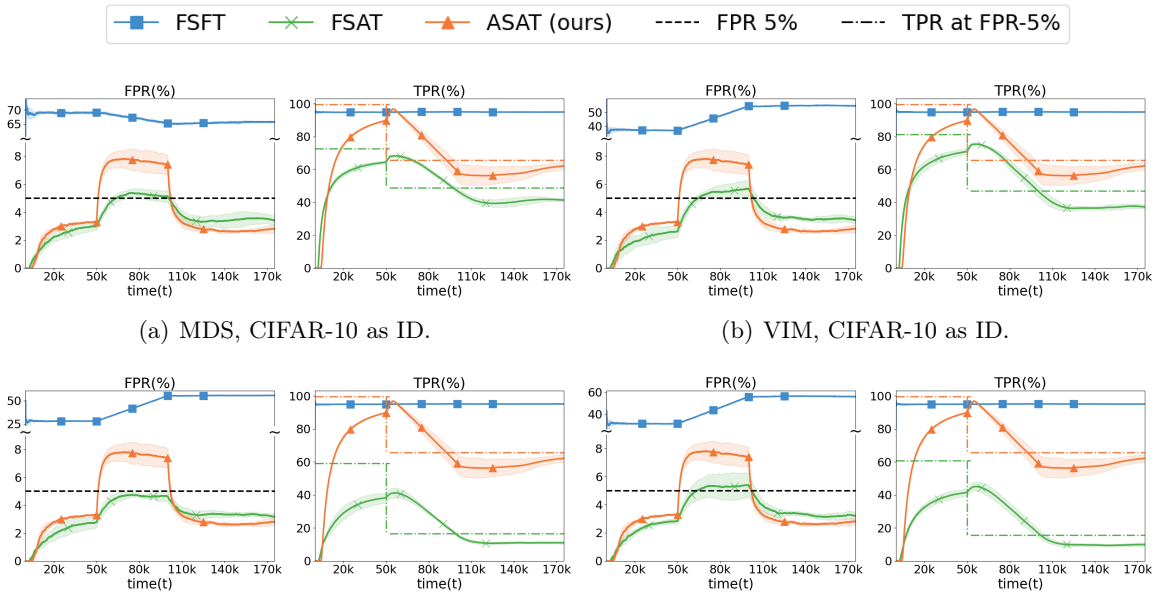
(c) VIM, CIFAR-10 as ID and CIFAR-100 as OOD. (d) VIM, CIFAR-10 as ID and Near-OOD as OOD.

(e) ODIN, CIFAR-10 as ID and CIFAR-100 as OOD. (f) ODIN, CIFAR-10 as ID and Near-OOD as OOD.

(g) ASH, CIFAR-10 as ID and CIFAR-100 as OOD. (h) ASH, CIFAR-10 as ID and Near-OOD as OOD.

(i) SHE, CIFAR-10 as ID and CIFAR-100 as OOD. (j) SHE, CIFAR-10 as ID and Near-OOD as OOD.

Figure F.3: Results for stationary settings using CIFAR-10 as ID, with each method repeated 5 times (mean and std shown). Run with five additional scoring functions, MDS, VIM, ODIN, ASH, and SHE.



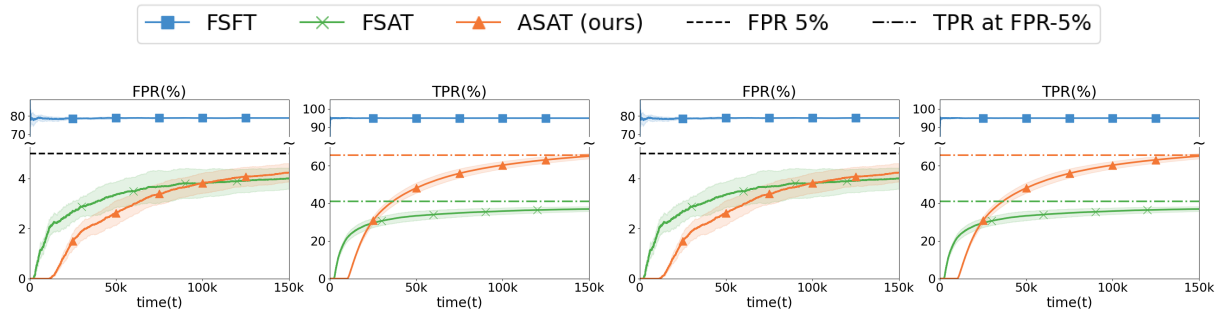
(a) MDS, CIFAR-10 as ID.

(b) VIM, CIFAR-10 as ID.

(c) ODIN, CIFAR-10 as ID.

(d) SHE, CIFAR-10 as ID.

Figure F.4: Results for nonstationary OOD settings using CIFAR-10 as ID, with each method repeated 5 times (mean and std shown). OOD distribution shifts from Far-OOD to Near-OOD. Run with four additional scoring functions, MDS, VIM, ODIN, and SHE.



(a) EBO, CIFAR-100 as ID and CIFAR-10 as OOD.

(b) EBO, CIFAR-100 as ID and Near-OOD as OOD.

(c) KNN, CIFAR-100 as ID and CIFAR-10 as OOD.

(d) KNN, CIFAR-100 as ID and Near-OOD as OOD.

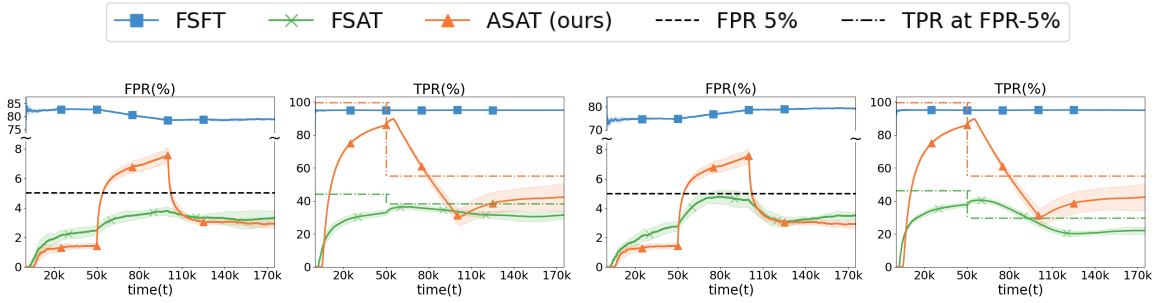
(e) ODIN, CIFAR-100 as ID and CIFAR-10 as OOD.

(f) ODIN, CIFAR-100 as ID and Near-OOD as OOD.

(g) ASH, CIFAR-100 as ID and CIFAR-10 as OOD.

(h) ASH, CIFAR-100 as ID and Near-OOD as OOD.

Figure F.5: Results for stationary settings using CIFAR-100 as ID, with each method repeated 5 times (mean and std shown). Run with four scoring functions, EBO, KNN, VIM, and ODIN.



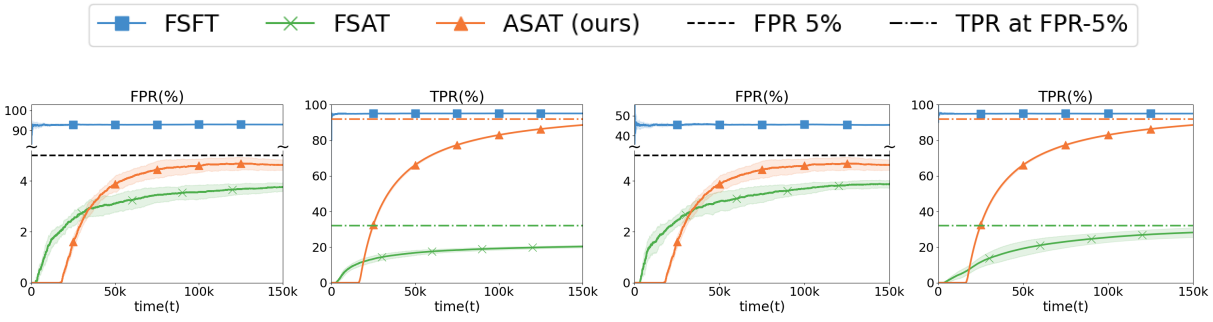
(a) EBO, CIFAR-100 as ID.

(b) KNN, CIFAR-100 as ID.

(c) ODIN, CIFAR-100 as ID.

(d) VIM, CIFAR-100 as ID.

Figure F.6: Results for nonstationary settings using CIFAR-100 as ID, with each method repeated 5 times (mean and std shown). OOD distribution shifts from Far-OOD to Near-OOD. Run with four additional scoring functions, MDS, VIM, ODIN, and SHE.



(a) Stationary, KNN, OpenImage-O as OOD.

(b) Stationary, MDS, OpenImage-O as OOD.

Figure F.7: Results on the stationary setting for ImageNet-1k as ID, with each method repeated 5 times (mean and std shown). KNN and MDS are the initial scoring functions and are fixed for FSFT and FSAT. The dotted-dash lines represent the *expected* TPR at FPR-5% for KNN, MDS, and $\widehat{g}^*$ (matching colors).

Appendix G

Appendix to Chapter 10

The appendix is organized as follows. In Section G.1.1, we provide details of LLM inference for MCQs. Additional experiments and results are given in Section G.2. First, in Section G.2.1 we discuss the trade-off between coverage (choice of α) in conformal prediction and its effect on CROQ accuracy. Next, in Section G.2.2, we explore the effectiveness of conformal prediction with CP-OPT scores in deferral applications. The Sections G.2.3,G.2.4, and G.2.5, contain more detailed results for the hypotheses discussed in the main paper. Section G.3 provides details of the procedure used to compute statistical significance. In Section G.4 we provide details of datasets and give samples of prompts before and after CROQ and LLM's answers. Finally, Section G.4.3 lists the hyperparameters used for learning the score function using CP-OPT.

G.1 Methodology and Background Details

G.1.1 Details on LLM inference in multi-choice question answering

We provide a formal description of the inference procedure described in the **LLM Inference** paragraph of Section 10.2.1.

The input prompt x is a sequence of tokens $t_1, t_2, \ldots t_n$. We run the forward pass of the auto-regressive LLM (Touvron et al., 2023; Dubey et al., 2024; Abdin et al., 2024) on x to produce a

set of output logits:

$$\boldsymbol{l}_1, \boldsymbol{l}_2, \ldots, \boldsymbol{l}_n \leftarrow \texttt{LLM}\big(t_1, t_2, \ldots t_n\big) \tag{G.1}$$

Here, each logit $\boldsymbol{l}_j \in \mathbb{R}^{|V|}$ expresses the likelihood of the next token after $t_1, \ldots, t_j$, where V is the universal set of tokens (aka the alphabet) for the given LLM and $|V|$ is its size. The last token's logits $\boldsymbol{l}_n$ are expected to have a high value for the correct answer key. We extract the logit vector $\bar{\boldsymbol{l}} \in \mathbb{R}^m$ corresponding to the option keys as follows:

$$\bar{\boldsymbol{l}} := \big[\, \boldsymbol{l}_n[Y_1], \boldsymbol{l}_n[Y_2], \ldots, \boldsymbol{l}_n[Y_m] \,\big], \tag{G.2}$$

where $\boldsymbol{l}_n[Y_j]$ denotes the logit value corresponding to the token Y_j in the last token's logits $\boldsymbol{l}_n$. The logits $\bar{\boldsymbol{l}}$ are converted to softmax scores $s(x)$. The softmax score of point x and option key y is denoted by $s(x, y)$ and the predicted answer key $\hat{y}$ corresponds to the maximum softmax value:

$$s(x) := \texttt{softmax}(\bar{\boldsymbol{l}}), \qquad s(x, y) := s(x)[y], \qquad \hat{y} := \underset{y \in \{Y_1, \ldots Y_m\}}{\arg\max}\ s(x, y) \tag{G.3}$$

G.2 Additional Experiments and Results

This appendix contains additional results and details not included in the main paper due to length constraints.

G.2.1 Trade-off between coverage and accuracy

The choice of α controls the coverage level in conformal prediction. A small α implies high coverage, meaning the prediction sets contain the true options with high probability but potentially have large sizes. Thus, choosing a very small α will likely not eliminate a sufficient number of options to see any noticeable improvement with CROQ. On the other hand, choosing a large α will eliminate the true option from the set for a large portion of the questions, which will result in low accuracy from CROQ. To study these trade-offs, we run CROQ with different values of α. The accuracy before and after CROQ for a range of α values are shown in Figure G.3 and

Figure G.2 for the Llama-3 and Phi-3 models respectively. The results are as expected given the observations above: using an overly conservative (small) α does not give much improvement; as we increase α, the accuracy also increases up to a point, after which it starts to come down. This suggests that to optimize accuracy, a practitioner can tune α for their chosen score function and setting.

G.2.2 Using conformal prediction for deferral

Smaller prediction sets imply fewer deferrals in human-in-the-loop or model cascade systems. We consider a deferral procedure in which a set size cutoff is selected, and the LLM answer is only retained if the set size is at or below that cutoff. For all larger sets, the question is passed to a human (or a more powerful but costly model) who can answer the question correctly. Smaller sets from CP are desirable for this procedure to be effective. We evaluate this procedure with logit and CP-OPT scores in two settings and show the results in Figure G.1. As expected, lower set size cutoffs result in higher accuracy. As the set size cutoff increases, the accuracy approaches the LLM's marginal accuracy, while the number of deferrals (i.e. the cost of obtaining the answer from a human or more expensive model) decreases. In the top row of the figure, the differences in the set sizes between logit and CP-OPT scores are not large enough to see a meaningful difference in this procedure. However, in the bottom row corresponding to the Gemma-2 model and TruthfulQA dataset with 15 options, we see CP-OPT scores lead to fewer deferrals in comparison to logits. Model cascades (Dohan et al., 2022; Gupta et al., 2024b) and deferrals to human-in-the-loop (Tailor et al., 2024b; Vishwakarma et al., 2024c) and more broadly selective prediction (El-Yaniv and Wiener, 2010; Fisch et al., 2022; Vishwakarma et al., 2023) are useful frameworks for model deployment while ensuring safety, high accuracy, and balancing the costs. Our experiments show the promise of CP with logit and CP-OPT scores in this task and suggest it would be fruitful to explore this design space with CP.

Figure G.2 shows accuracy after the CROQ procedure as a function of α for Phi-3. The results are qualitatively similar to the results for Llama-3 in the main text (Section 10.4.2).

All remaining results are organized by dataset. Tables for the CROQ results, which illustrate

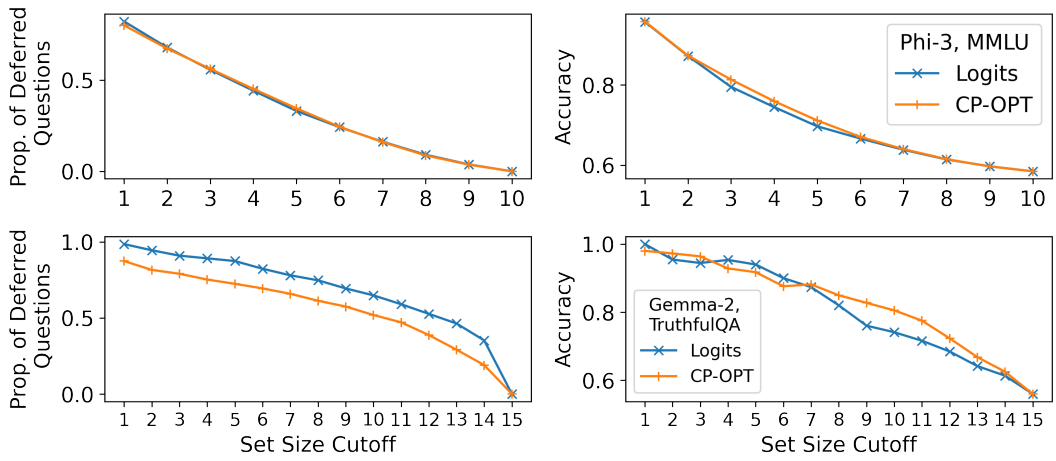


Figure G.1: Proportion of questions deferred to a human when conformal prediction set sizes exceed a certain cutoff (left), and the corresponding LLM accuracy for questions (without revision) retained by the LLM as a function of cutoff threshold (right). In the top row (MMLU, 10 options, `Phi-3-4k-mini-Instruct`), the difference in deferral and accuracy is negligible, whereas in the bottom row (TruthfulQA, 15 options, `gemma-2-9b-it-SimPO`), CP-OPT defers fewer questions to the human while providing similar or improved accuracy for questions retained.

accuracy changes conditional on set size, are based on a confidence level of 95% (equivalently, an α level of 0.05). Note that with the ToolAlpaca dataset, not all possible set sizes occur, in which case we omit the corresponding columns. For example, with 10 response options, only sets of size 8 and smaller occur.

Asterisks in the tables indicate where the difference in overall accuracy from Before to After, i.e., from baseline to after the CROQ procedure, is statistically significant at the $\alpha = 0.05$ level. (In some tables, like Table G.5, none of the changes are significant.) See Appendix G.3 for details on how statistical significance was calculated.

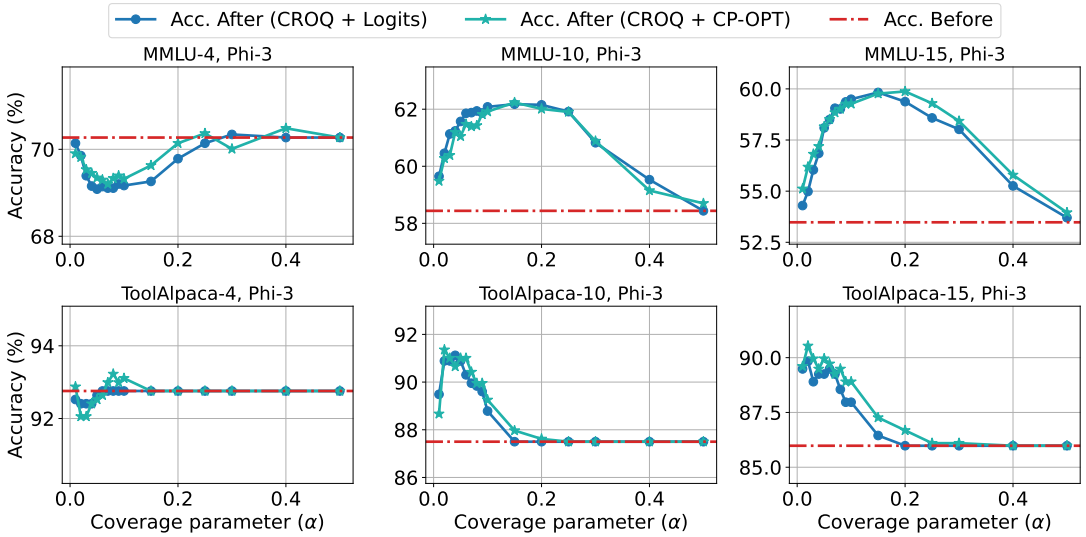


Figure G.2: Accuracy on revised questions on the MMLU and ToolAlpaca datasets while varying miscoverage parameter α for `Phi-3-4k-mini-Instruct` (Phi-3) model and both scores. Smaller values of α correspond to high levels of coverage. When coverage is too large, few or no answers are eliminated, and the LLM is prompted with the same question. When coverage is low, a larger portion of answer sets no longer contain the true answer, and the benefits of revision are diminished.

G.2.3 MMLU

Results for the experiments on the MMLU dataset are given in Tables G.5 and G.6,Tables G.2 to G.4 and Figs. G.4 to G.6.

G.2.4 TruthfulQA

Results for the experiments on the TruthfulQA dataset are given in Tables G.7 to G.11 and Figs. G.11 and G.12.

G.2.5 ToolAlpaca

Results for experiments on the ToolAlpaca dataset are given in Tables G.12 to G.16 and Figs. G.8 and G.9.

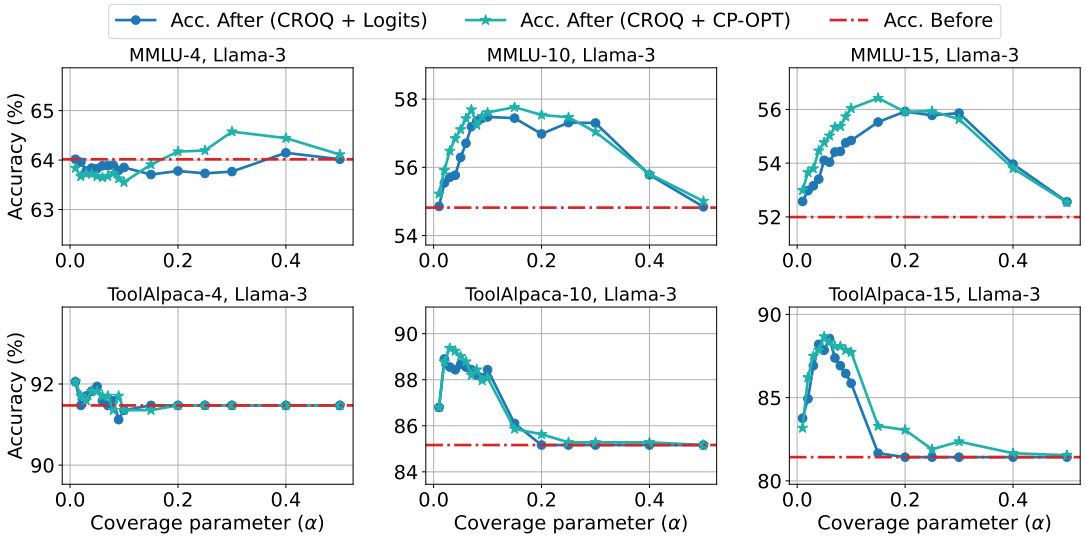
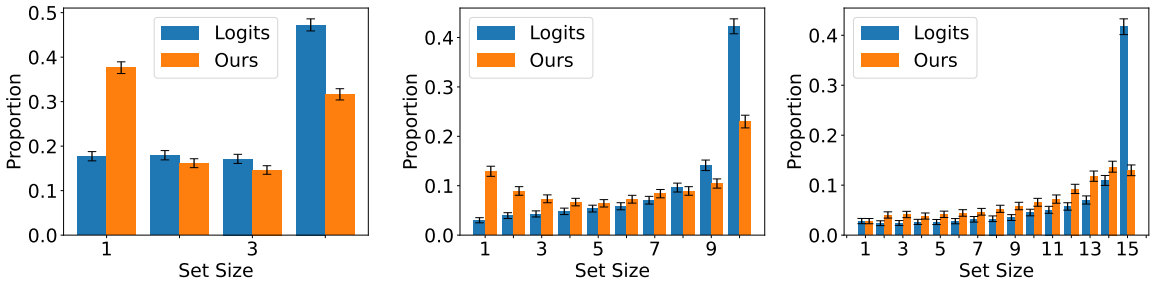
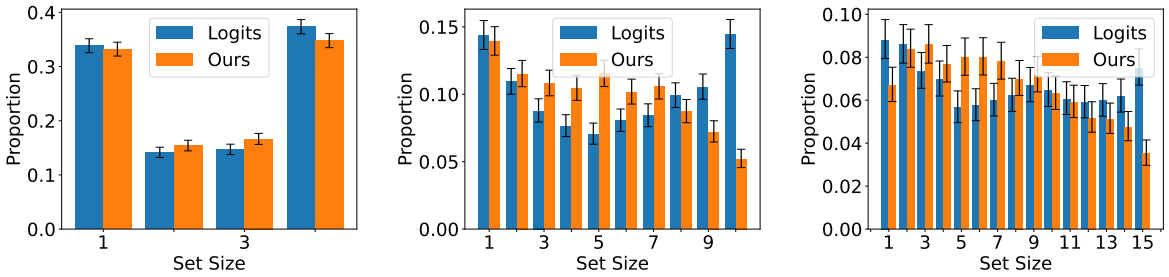


Figure G.3: Accuracy on revised questions on the MMLU and ToolAlpaca datasets while varying miscoverage parameter α for `Llama-3-8B-Instruct` (Llama-3) model and both scores. Smaller values of α correspond to high levels of coverage. When coverage is too large, few or no answers are eliminated, and the LLM is prompted with the same question. When coverage is low, a larger portion of answer sets no longer contain the true answer or produce empty prediction sets, thus resulting in diminished benefits of revision.



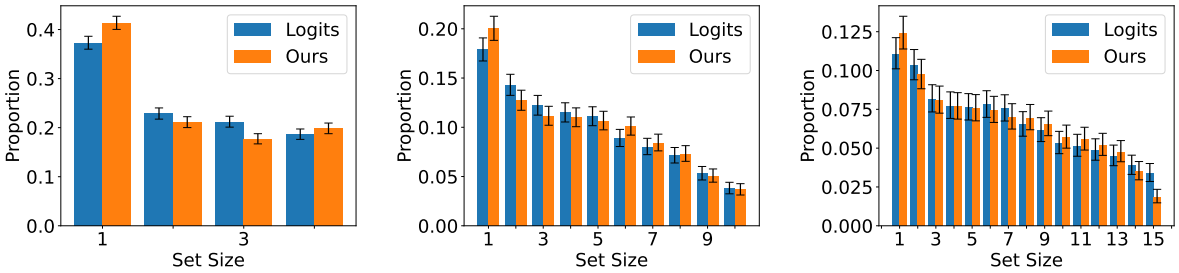
(a) MMLU 4 options. (b) MMLU 10 options. (c) MMLU 15 options.

Figure G.4: Distributions of sizes of sets obtained from CP-OPT and logit scores on MMLU dataset and Gemma-2 model.



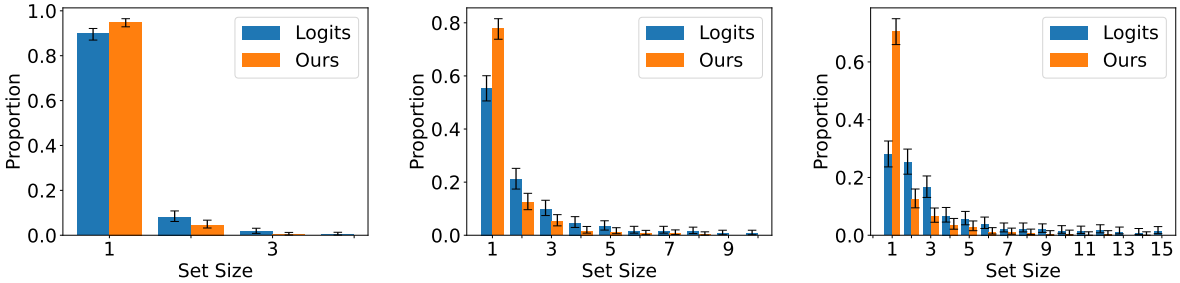
(a) MMLU 4 options. (b) MMLU 10 options. (c) MMLU 15 options.

Figure G.5: Distributions of sizes of sets obtained from CP-OPT and logit scores on MMLU dataset and Llama-3 model.



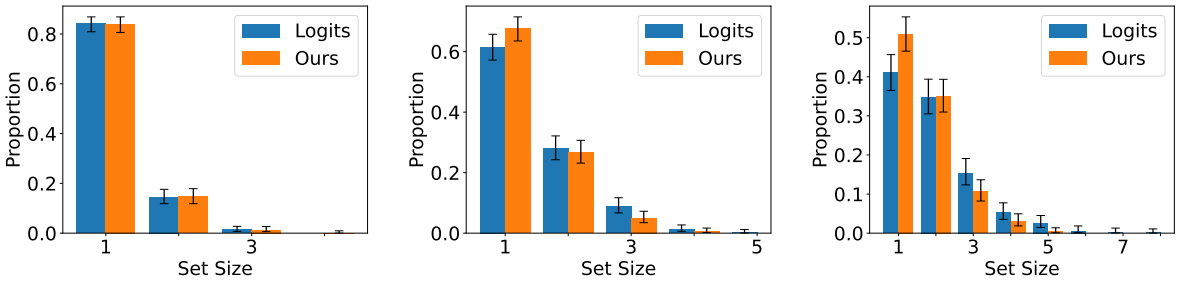
(a) MMLU 4 options. (b) MMLU 10 options. (c) MMLU 15 options.

Figure G.6: Distributions of sizes of sets obtained from CP-OPT and logit scores on MMLU dataset and Phi-3 model setting.



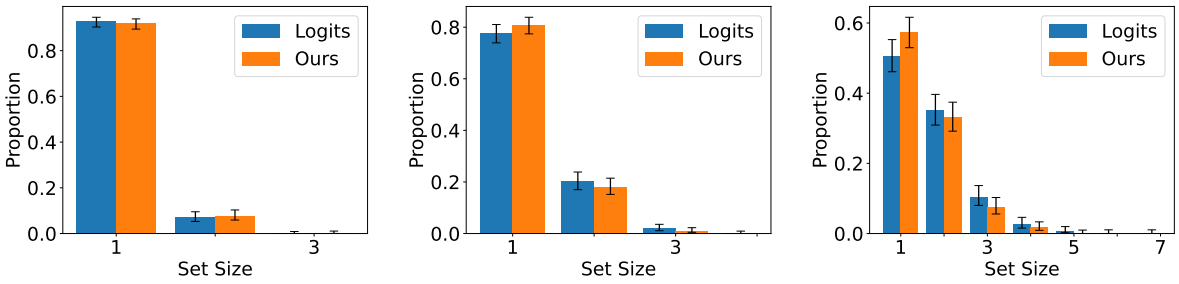
(a) ToolAlpaca 4 options. (b) ToolAlpaca 10 options. (c) ToolAlpaca 15 options.

Figure G.7: Distributions of sizes of sets obtained from CP-OPT and logit scores on ToolAlpaca dataset and Gemma-2 model.



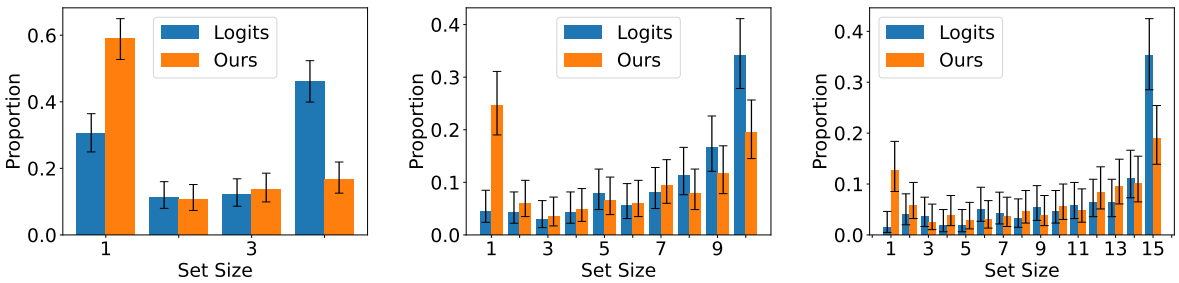
(a) ToolAlpaca 4 options. (b) ToolAlpaca 10 options. (c) ToolAlpaca 15 options.

Figure G.8: Distributions of sizes of sets obtained from CP-OPT and logit scores on ToolAlpaca dataset and Llama-3 model.



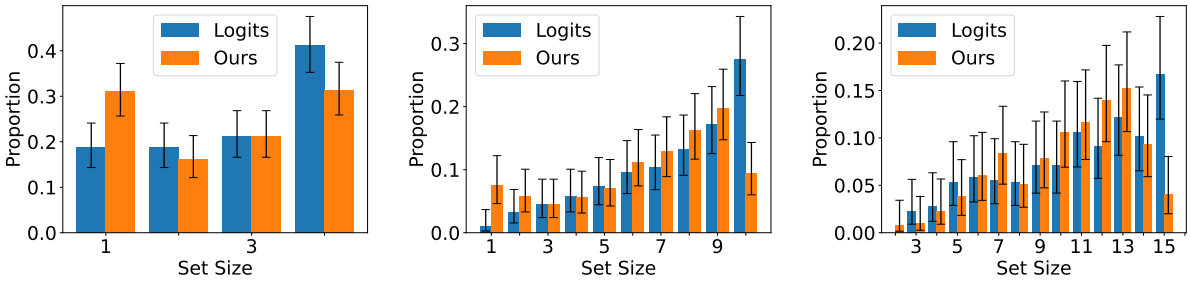
(a) ToolAlpaca 4 options. (b) ToolAlpaca 10 options. (c) ToolAlpaca 15 options.

Figure G.9: Distributions of sizes of sets obtained from CP-OPT and logit scores on ToolAlpaca dataset and Phi-3 model.



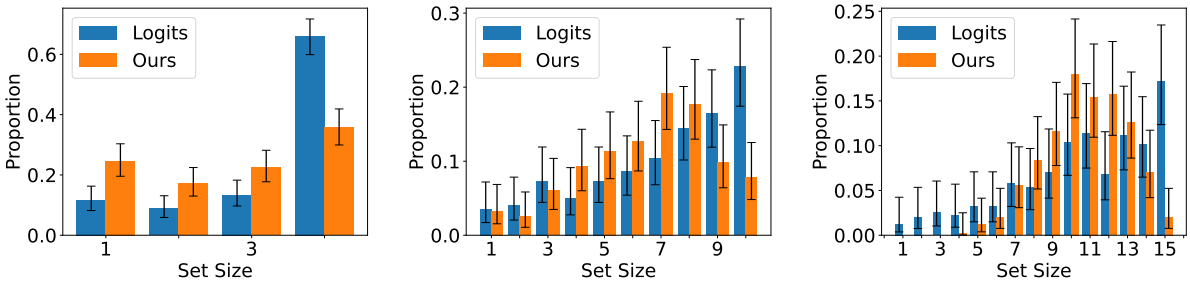
(a) Truthful QA 4 options. (b) Truthful QA 10 options. (c) Truthful QA 15 options.

Figure G.10: Distributions of sizes of sets obtained from CP-OPT and logit scores on Truthful QA dataset and Gemma-2.



(a) Truthful QA 4 options. (b) Truthful QA 10 options. (c) Truthful QA 15 options.

Figure G.11: Distributions of sizes of sets obtained from CP-OPT and logit scores on Truthful QA dataset and Phi-3 model.



(a) Truthful QA 4 options. (b) Truthful QA 10 options. (c) Truthful QA 15 options.

Figure G.12: Distributions of sizes of sets obtained from CP-OPT and logit scores on Truthful QA dataset and Llama-3 model.

Model	# Opt.	LLama-3			Phi-3			Gemma-2		
		Accuracy Before (a_2)	Accuracy After (a_2')	Gain ($a_2' - a_2$)	Accuracy Before (a_2)	Accuracy After (a_2')	Gain ($a_2' - a_2$)	Accuracy Before (a_2)	Accuracy After (a_2')	Gain ($a_2' - a_2$)
MMLU	4	**64.02**	63.67	-0.34	**70.27**	69.34	-0.93	68.36	**69.56**	**1.20***
	10	54.82	**57.11**	**2.29***	58.44	**61.05**	**2.61***	53.99	**57.93**	**3.94***
	15	51.99	**54.77**	**2.78***	53.48	**58.15**	**4.68***	50.78	**51.31**	**0.52**
ToolAlpaca	4	91.47	**91.82**	**0.35**	**92.64**	92.52	-0.12	93.46	**93.57**	**0.12**
	10	85.16	**89.02**	**3.86***	87.62	**91.00**	**3.39***	88.08	**90.42**	**2.34***
	15	81.43	**88.67**	**7.24***	85.98	**89.95**	**3.97***	88.08	**89.37**	**1.29**
TruthfulQA	4	54.43	**55.44**	**1.01**	69.87	**69.87**	**0.00**	74.94	**76.96**	**2.03**
	10	39.24	**42.28**	**3.04**	55.70	**56.20**	**0.51**	56.46	**60.76**	**4.30***
	15	37.22	**37.47**	**0.25**	46.84	**51.39**	**4.56***	55.95	**57.72**	**1.77**

Table G.1: [CROQ + CP-OPT]. Results on accuracy improvement with CROQ using CP-OPT scores. Here, a_2, and a_2' refer to the accuracy before CROQ and after CROQ, respectively. A higher gain in a setting suggests CROQ improved accuracy in that setting.

Score	Set Size	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	Overall
Logits	**Coverage**	82.40	69.04	80.00	83.56	81.11	87.45	86.31	88.60	90.75	90.45	94.80	93.75	98.30	98.15	100	94.58
	Fraction	2.77	2.34	2.37	2.60	2.58	2.74	3.12	3.23	3.47	4.47	5.02	5.70	6.99	10.91	41.70	100
	Acc. Before	82.40	62.44	62.00	65.30	60.37	**61.47**	**61.98**	**59.19**	55.82	**62.6**	**57.92**	51.25	57.89	**50.38**	40.01	**50.78**
	Acc. After	**82.40**	**65.48**	**68.50**	**65.75**	**63.13**	58.87	60.08	57.72	**56.85**	58.89	55.08	**51.88**	**58.06**	49.40	**40.01**	50.58
Ours	**Coverage**	93.10	94.05	89.83	89.94	89.34	90.54	89.74	90.23	92.40	94.73	94.70	94.46	96.77	97.74	100	94.58
	Fraction	2.75	3.99	4.08	3.77	4.12	4.39	4.63	5.22	5.78	6.53	7.17	9.21	11.76	13.66	12.94	100
	Acc. Before	93.10	88.10	82.56	79.56	**75.79**	**73.24**	**64.62**	**56.82**	56.26	52.73	45.20	42.53	36.63	33.10	25.96	50.78
	Acc. After	**93.10**	**89.58**	**82.56**	**80.82**	73.78	70.81	60.26	56.14	**57.49**	**53.27**	**46.69**	**43.94**	**40.06**	**33.80**	**25.96**	**51.31**

Table G.2: Results for CROQ on the MMLU dataset with 15 response options and Gemma-2 model.

Score	Set Size	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	Overall
Logits	**Coverage**	95.82	91.56	89.98	93.19	94.54	94.63	94.44	95.60	96.09	96.88	97.06	96.77	98.21	98.08	100	95.42
	Fraction	8.81	8.58	7.35	6.97	5.65	5.74	5.98	6.21	6.68	6.46	6.05	5.89	5.97	6.17	7.50	100
	Acc. Before	95.82	82.16	72.37	66.95	55.88	50.62	50.20	**46.08**	40.14	37.32	34.90	34.68	30.62	**27.88**	24.05	51.99
	Acc. After	**95.82**	**83.82**	**76.09**	**71.55**	**63.66**	**53.93**	**51.39**	45.32	**43.69**	**40.99**	**36.47**	**35.08**	**33.00**	27.69	**24.05**	**54.11***
Ours	**Coverage**	94.15	94.62	91.29	91.63	93.31	93.18	94.52	96.43	97.02	96.42	97.59	96.56	97.91	98.25	100	95.06
	Fraction	6.69	8.38	8.58	7.65	7.99	8.00	7.80	6.99	7.17	6.30	5.90	5.17	5.12	4.75	3.51	100
	Acc. Before	94.15	87.54	73.58	65.58	55.57	51.78	45.81	46.86	39.90	31.83	33.00	28.67	**31.32**	21.25	19.59	51.99
	Acc. After	**94.15**	**89.24**	**75.80**	**70.39**	**63.74**	**54.60**	**50.53**	**47.54**	**42.38**	**35.03**	**34.21**	**33.26**	29.93	**24.75**	**19.59**	**54.77***

Table G.3: Results for CROQ on the MMLU dataset with 15 response options and Llama-3 model.

Score	Set Size	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	Overall
Logits	**Coverage**	96.03	92.77	93.46	91.71	93.93	93.61	93.55	93.81	94.79	96.65	95.38	96.83	95.77	97.25	100	94.60
	Fraction	11.07	10.34	8.17	7.73	7.62	7.80	7.55	6.52	6.15	5.32	5.14	4.87	4.49	3.88	3.38	100
	Acc. Before	96.03	80.48	69.62	59.14	53.12	46.27	42.61	42.08	37.84	39.51	36.72	34.15	23.02	23.55	21.75	53.48
	Acc. After	**96.03**	**84.85**	**76.60**	**66.97**	**63.86**	**53.42**	**51.10**	**44.44**	**42.86**	**42.19**	**39.26**	**36.34**	**25.13**	**24.46**	**21.75**	**58.09***
Ours	**Coverage**	95.79	92.20	93.83	91.19	94.19	93.79	95.93	94.54	94.57	96.04	93.82	96.80	96.26	97.29	100	94.61
	Fraction	12.40	9.73	8.08	7.68	7.56	7.45	7.00	6.95	6.55	5.70	5.57	5.20	4.76	3.50	1.86	100
	Acc. Before	95.79	80.24	73.86	60.28	51.33	49.68	43.90	41.47	36.41	31.46	29.42	29.00	25.69	21.69	18.47	53.48
	Acc. After	**95.79**	**83.66**	**78.12**	**69.86**	**62.64**	**54.62**	**52.03**	**47.95**	**39.67**	**38.96**	**32.41**	**31.28**	**27.18**	**22.37**	**18.47**	**58.15***

Table G.4: Results for CROQ on the MMLU dataset with 15 response options and Phi-3 model.

Model	Score	Set Size	1	2	3	4	Overall
Gemma-2	Logits	**Coverage**	89.34	89.94	93.27	100	95.16
		Fraction	17.71	17.93	17.11	47.25	100
		Acc. Before	89.34	79.42	**68.24**	54.79	67.62
		Acc. After	**89.34**	**79.95**	68.10	**54.79**	**67.70**
	Ours	**Coverage**	91.67	89.93	93.10	100	94.23
		Fraction	37.62	16.14	14.61	31.63	100
		Acc. Before	91.67	72.50	57.27	43.64	68.36
		Acc. After	**91.67**	**75.88**	**61.74**	**43.64**	**69.56***
Llama-3	Logits	**Coverage**	93.55	92.78	92.89	100	95.75
		Fraction	33.84	14.13	14.68	37.35	100
		Acc. Before	93.55	70.19	**49.88**	40.48	**64.02**
		Acc. After	**93.55**	**70.70**	48.10	**40.48**	63.83
	Ours	**Coverage**	93.71	91.83	93.50	100	95.57
		Fraction	33.21	15.39	16.63	34.77	100
		Acc. Before	93.71	**71.16**	**52.46**	38.02	**64.02**
		Acc. After	**93.71**	70.01	51.46	**38.02**	63.67
Phi-3	Logits	**Coverage**	94.75	91.48	93.17	100	94.65
		Fraction	37.30	22.86	21.20	18.64	100
		Acc. Before	94.75	**70.25**	**52.69**	41.31	**70.27**
		Acc. After	**94.75**	66.93	50.67	**41.31**	69.08
	Ours	**Coverage**	93.63	90.61	94.17	100	94.35
		Fraction	41.36	21.10	17.71	19.83	100
		Acc. Before	93.63	**67.38**	**52.82**	40.22	**70.27**
		Acc. After	**93.63**	64.57	50.94	**40.22**	69.34

Table G.5: Results for CROQ on the MMLU dataset with 4 response options.

Model	Score	Set Size	1	2	3	4	5	6	7	8	9	10	Overall
Gemma-2	Logits	**Coverage**	78.80	79.03	84.92	88.56	85.30	92.64	94.09	96.41	97.22	100	95.00
		Fraction	2.97	3.90	4.25	4.77	5.33	5.80	7.03	9.59	14.10	42.26	100
		Acc. Before	78.80	73.86	74.02	68.41	62.36	**67.69**	**61.49**	**58.42**	**51.94**	41.81	53.80
		Acc. After	**78.80**	**76.90**	**75.98**	**72.39**	**62.36**	66.67	60.14	57.67	51.68	**41.81**	**53.93**
	Ours	**Coverage**	90.79	92.27	88.31	90.54	89.80	91.30	92.05	95.60	97.49	100	94.04
		Fraction	12.89	8.90	7.31	6.65	6.40	7.23	8.36	8.90	10.41	22.96	100
		Acc. Before	90.79	84.93	69.97	66.07	54.17	48.60	42.76	40.00	37.74	31.27	53.99
		Acc. After	**90.79**	**89.20**	**79.87**	**75.00**	**64.01**	**55.34**	**47.02**	**45.33**	**40.59**	**31.27**	**57.93***
Llama-3	Logits	**Coverage**	94.55	91.96	91.73	94.09	94.94	97.19	97.32	97.72	99.32	100	96.06
		Fraction	14.36	10.92	8.76	7.63	7.04	8.03	8.40	9.90	10.53	14.43	100
		Acc. Before	94.55	**80.43**	65.99	57.54	51.43	47.56	37.71	35.13	34.84	31.41	54.82
		Acc. After	**94.55**	80.33	**69.51**	**60.96**	**53.29**	**49.93**	**42.37**	**36.21**	**35.74**	**31.41**	**56.29***
	Ours	**Coverage**	94.80	91.95	92.42	93.98	94.95	96.61	97.64	97.96	98.68	100	95.45
		Fraction	13.92	11.50	10.80	10.44	11.51	10.16	10.55	8.71	7.20	5.20	100
		Acc. Before	94.80	**79.67**	68.02	52.61	45.05	40.19	35.55	33.65	28.67	30.82	54.82
		Acc. After	**94.80**	79.05	**71.76**	**55.57**	**49.90**	**42.76**	**40.83**	**35.42**	**30.31**	**30.82**	**57.11***
Phi-3	Logits	**Coverage**	95.75	91.02	90.76	94.21	93.90	95.59	94.07	96.17	95.52	100	94.11
		Fraction	17.87	14.28	12.20	11.48	11.08	8.88	8.01	7.12	5.29	3.79	100
		Acc. Before	95.75	76.56	59.14	55.02	45.50	43.72	37.19	**33.0**	30.27	26.65	58.44
		Acc. After	**95.75**	**79.05**	**65.56**	**59.77**	**51.18**	**47.19**	**42.37**	32.83	**32.29**	**26.65**	**61.57***
	Ours	**Coverage**	95.85	90.94	90.94	94.05	93.53	94.71	93.94	94.96	96.71	100	94.09
		Fraction	20.02	12.71	11.13	10.98	10.65	10.09	8.41	7.30	5.06	3.66	100
		Acc. Before	95.85	73.86	63.75	54.38	46.38	40.47	36.53	32.68	**26.76**	26.30	58.44
		Acc. After	**95.85**	**76.84**	**68.66**	**59.68**	**50.61**	**44.12**	**38.50**	**34.80**	26.06	**26.30**	**61.05***

Table G.6: Results for CROQ on the MMLU dataset with 10 response options.

Score	Set Size	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	Overall
Logits	**Coverage**	100	93.75	92.86	100	100	95.00	94.12	76.92	80.95	94.44	100	88.00	88.00	100	100	95.44
	Fraction	1.52	4.05	3.54	1.77	1.77	5.06	4.30	3.29	5.32	4.56	5.82	6.33	6.33	11.14	35.19	100
	Acc. Before	100	93.75	92.86	100	85.71	80.00	76.47	46.15	47.62	**61.11**	**56.52**	48.00	32.00	**47.73**	46.04	55.95
	Acc. After	**100**	**93.75**	**92.86**	**100**	**85.71**	**85.00**	**82.35**	**53.85**	**57.14**	55.56	52.17	**48.00**	**40.00**	45.45	**46.04**	**56.96**
Ours	**Coverage**	98.00	95.65	90.00	93.33	90.91	91.67	92.86	94.44	93.33	95.45	89.47	96.97	97.37	100	100	96.46
	Fraction	12.66	5.82	2.53	3.80	2.78	3.04	3.54	4.56	3.80	5.57	4.81	8.35	9.62	10.13	18.99	100
	Acc. Before	98.00	**95.65**	90.00	73.33	81.82	50.00	92.86	61.11	60.00	63.64	47.37	39.39	31.58	32.50	28.00	55.95
	Acc. After	**98.00**	91.30	**90.00**	**80.00**	**81.82**	**58.33**	**92.86**	**61.11**	**60.00**	**72.73**	**52.63**	**42.42**	**36.84**	**32.50**	**28.00**	**57.72**

Table G.7: Results for CROQ on the TruthfulQA dataset with 15 response options and Gemma-2 model

Score	Set Size	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	Overall
Logits	**Coverage**	80.00	75.00	90.00	77.78	76.92	76.92	86.96	95.24	100	95.12	100	92.59	97.73	100	100	94.68
	Fraction	1.27	2.03	2.53	2.28	3.29	3.29	5.82	5.32	7.09	10.38	11.39	6.84	11.14	10.13	17.22	100
	Acc. Before	80.00	62.50	80.00	66.67	53.85	38.46	60.87	**57.14**	**50.0**	**46.34**	31.11	29.63	**22.73**	15.00	22.06	37.22
	Acc. After	**80.00**	**75.00**	**90.00**	**66.67**	**61.54**	**38.46**	**60.87**	52.38	46.43	43.90	**33.33**	**29.63**	18.18	**17.50**	**22.06**	**37.22**
Ours	**Coverage**	0	0	0	0	100	87.50	81.82	93.94	91.30	94.37	100	95.16	96.00	100	100	94.68
	Fraction	0	0	0	0.25	1.27	2.03	5.57	8.35	11.65	17.97	15.44	15.70	12.66	7.09	2.03	100
	Acc. Before	0	0	0	0	80.00	37.50	40.91	60.61	28.26	**45.07**	**44.26**	32.26	22.00	28.57	0	37.22
	Acc. After	**0**	**0**	**0**	**0**	**80.00**	**50.00**	**40.91**	**60.61**	**36.96**	36.62	42.62	**32.26**	**26.00**	**32.14**	**0**	**37.47**

Table G.8: Results for CROQ on the TruthfulQA dataset with 15 response options and Llama-3.

Score	Set Size	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	Overall
Logits	**Coverage**	0	0	88.89	90.91	85.71	82.61	95.45	85.71	96.43	100	92.86	100	100	97.50	100	95.44
	Fraction	0	0	2.28	2.78	5.32	5.82	5.57	5.32	7.09	7.09	10.63	9.11	12.15	10.13	16.71	100
	Acc. Before	0	0	77.78	90.91	52.38	56.52	63.64	**61.9**	**60.71**	50.00	**35.71**	**33.33**	50.00	**30.0**	34.85	**46.84**
	Acc. After	**0**	**0**	**77.78**	**90.91**	**52.38**	**60.87**	**63.64**	57.14	57.14	**57.14**	33.33	27.78	**52.08**	27.50	**34.85**	46.33
Ours	**Coverage**	0	100	100	88.89	93.33	91.67	100	85.00	96.77	95.24	95.65	98.18	98.33	100	100	96.46
	Fraction	0	0.76	1.01	2.28	3.80	6.08	8.35	5.06	7.85	10.63	11.65	13.92	15.19	9.37	4.05	100
	Acc. Before	0	100	100	77.78	60.00	62.50	66.67	45.00	58.06	45.24	47.83	36.36	30.00	37.84	31.25	46.84
	Acc. After	**0**	**100**	**100**	**77.78**	**66.67**	**62.50**	**72.73**	**45.00**	**58.06**	**57.14**	**50.00**	**43.64**	**36.67**	**40.54**	**31.25**	**51.39***

Table G.9: Results for CROQ on the TruthfulQA dataset with 15 response options and Phi-3 model.

Model	Score	Set Size	1	2	3	4	5	6	7	8	9	10	Overall
Gemma-2	Logits	**Coverage**	100	94.12	100	94.12	87.10	90.91	90.62	91.11	95.45	100	95.44
		Fraction	4.56	4.30	3.04	4.30	7.85	5.57	8.10	11.39	16.71	34.18	100
		Acc. Before	100	94.12	100	82.35	70.97	**63.64**	56.25	**53.33**	53.03	37.04	**56.46**
		Acc. After	**100**	**94.12**	**100**	**82.35**	**70.97**	59.09	**56.25**	51.11	**54.55**	**37.04**	56.20
	Ours	**Coverage**	97.94	100	92.86	89.47	96.15	91.67	100	93.55	97.83	100	97.22
		Fraction	24.56	6.08	3.54	4.81	6.58	6.08	9.37	7.85	11.65	19.49	100
		Acc. Before	97.94	91.67	**85.71**	52.63	61.54	66.67	54.05	19.35	32.61	14.29	56.46
		Acc. After	**97.94**	**95.83**	71.43	**89.47**	**73.08**	**66.67**	**59.46**	**29.03**	**39.13**	**14.29**	**60.76***
Llama-3	Logits	**Coverage**	92.86	93.75	68.97	95.00	86.21	91.18	97.56	96.49	100	100	94.43
		Fraction	3.54	4.05	7.34	5.06	7.34	8.61	10.38	14.43	16.46	22.78	100
		Acc. Before	92.86	81.25	55.17	55.00	51.72	**41.18**	**41.46**	26.32	30.77	23.33	39.24
		Acc. After	**92.86**	**87.50**	**55.17**	**65.00**	**58.62**	38.24	34.15	**31.58**	**33.85**	**23.33**	**40.76**
	Ours	**Coverage**	92.31	90.00	70.83	91.89	95.56	92.00	92.11	97.14	100	100	93.42
		Fraction	3.29	2.53	6.08	9.37	11.39	12.66	19.24	17.72	9.87	7.85	100
		Acc. Before	92.31	70.00	54.17	56.76	51.11	44.00	**31.58**	28.57	20.51	16.13	39.24
		Acc. After	**92.31**	**80.00**	**58.33**	**72.97**	**55.56**	**50.00**	30.26	**28.57**	**20.51**	**16.13**	**42.28**
Phi-3	Logits	**Coverage**	100	100	94.44	100	96.55	89.47	100	100	100	100	98.48
		Fraction	1.01	3.29	4.56	5.82	7.34	9.62	10.38	13.16	17.22	27.59	100
		Acc. Before	100	100	83.33	69.57	65.52	55.26	**60.98**	51.92	**50.0**	42.20	**55.70**
		Acc. After	**100**	**100**	**88.89**	**69.57**	**65.52**	**55.26**	51.22	**51.92**	47.06	**42.20**	54.43
	Ours	**Coverage**	100	86.96	88.89	90.91	85.71	95.45	96.08	100	97.44	100	95.70
		Fraction	7.59	5.82	4.56	5.57	7.09	11.14	12.91	16.20	19.75	9.37	100
		Acc. Before	100	78.26	**83.33**	72.73	53.57	**65.91**	49.02	45.31	43.59	24.32	55.70
		Acc. After	**100**	**78.26**	77.78	**72.73**	**60.71**	61.36	**52.94**	**45.31**	**44.87**	**24.32**	**56.20**

Table G.10: Results for CROQ on the TruthfulQA dataset with 10 response options.

Model	Score	Set Size	1	2	3	4	Overall
Gemma-2	Logits	**Coverage**	95.00	93.33	89.58	100	96.46
		Fraction	30.38	11.39	12.15	46.08	100
		Acc. Before	95.00	84.44	68.75	60.44	74.68
		Acc. After	**95.00**	**86.67**	**68.75**	**60.44**	**74.94**
	Ours	**Coverage**	97.00	90.48	87.04	100	95.44
		Fraction	58.99	10.63	13.67	16.71	100
		Acc. Before	97.00	59.52	44.44	31.82	74.94
		Acc. After	**97.00**	**66.67**	**53.70**	**31.82**	**76.96**
Llama-3	Logits	**Coverage**	91.30	85.71	86.79	100	95.95
		Fraction	11.65	8.86	13.42	66.08	100
		Acc. Before	91.30	74.29	67.92	42.53	54.43
		Acc. After	**91.30**	**82.86**	**67.92**	**42.53**	**55.19**
	Ours	**Coverage**	90.72	82.35	89.89	100	92.41
		Fraction	24.56	17.22	22.53	35.70	100
		Acc. Before	90.72	60.29	42.70	34.04	54.43
		Acc. After	**90.72**	**63.24**	**44.94**	**34.04**	**55.44**
Phi-3	Logits	**Coverage**	98.65	90.54	94.05	100	96.71
		Fraction	18.73	18.73	21.27	41.27	100
		Acc. Before	98.65	**83.78**	65.48	52.76	69.87
		Acc. After	**98.65**	81.08	**69.05**	**52.76**	**70.13**
	Ours	**Coverage**	96.75	95.31	92.86	100	96.71
		Fraction	31.14	16.20	21.27	31.39	100
		Acc. Before	96.75	**82.81**	58.33	44.35	69.87
		Acc. After	**96.75**	81.25	**59.52**	**44.35**	**69.87**

Table G.11: Results for CROQ on the TruthfulQA dataset with 4 response options.

Model	Score	Set Size	1	2	3	4	Overall
Gemma-2	Logits	**Coverage**	95.71	95.71	92.86	100	95.68
		Fraction	89.84	8.18	1.64	0.35	100
		Acc. Before	95.71	**74.29**	**78.57**	33.33	**93.46**
		Acc. After	**95.71**	71.43	71.43	**33.33**	93.11
	Ours	**Coverage**	95.45	95.00	100	0	95.44
		Fraction	94.98	4.67	0.35	0	100
		Acc. Before	95.45	57.50	33.33	0	93.46
		Acc. After	**95.45**	**57.50**	**66.67**	**0**	**93.57**
Llama-3	Logits	**Coverage**	96.81	98.39	100	0	97.08
		Fraction	84.11	14.49	1.40	0	100
		Acc. Before	96.81	62.90	66.67	0	91.47
		Acc. After	**96.81**	**66.13**	**66.67**	**0**	**91.94**
	Ours	**Coverage**	96.66	97.60	100	100	96.85
		Fraction	84.00	14.60	1.29	0.12	100
		Acc. Before	96.66	64.00	**63.64**	100	91.47
		Acc. After	**96.66**	**68.80**	36.36	**100**	**91.82**
Phi-3	Logits	**Coverage**	95.47	93.44	100	0	95.33
		Fraction	92.76	7.13	0.12	0	100
		Acc. Before	95.47	**59.02**	0	0	**92.76**
		Acc. After	**95.47**	55.74	**100**	**0**	92.64
	Ours	**Coverage**	95.81	94.03	100	0	95.68
		Fraction	91.94	7.83	0.23	0	100
		Acc. Before	95.81	**56.72**	50.00	0	**92.64**
		Acc. After	**95.81**	55.22	**50.00**	**0**	92.52

Table G.12: Results for CROQ on the ToolAlpaca dataset with 4 response options.

Model	Score	Set Size	1	2	3	4	5	6	7	8	9	10	Overall
Gemma-2	Logits	**Coverage**	96.41	91.67	96.47	97.44	96.43	100	92.86	100	100	100	95.56
		Fraction	55.37	21.03	9.93	4.56	3.27	1.64	1.64	1.40	0.58	0.58	100
		Acc. Before	96.41	85.56	78.82	69.23	82.14	50.00	35.71	50.00	80.00	20.00	87.73
		Acc. After	**96.41**	**86.67**	**87.06**	**71.79**	**85.71**	**71.43**	**42.86**	**58.33**	**80.00**	**20.00**	**89.60***
	Ours	**Coverage**	95.05	94.34	91.11	78.57	90.91	100	100	100	0	0	94.51
		Fraction	77.92	12.38	5.26	1.64	1.29	0.58	0.70	0.23	0	0	100
		Acc. Before	95.05	73.58	57.78	35.71	45.45	20.00	50.00	100	0	0	88.08
		Acc. After	**95.05**	**80.19**	**68.89**	**64.29**	**72.73**	**40.00**	**50.00**	**100**	**0**	**0**	**90.42***
Llama-3	Logits	**Coverage**	95.64	94.17	94.74	100	100	0	0	0	0	0	95.21
		Fraction	61.57	28.04	8.88	1.29	0.23	0	0	0	0	0	100
		Acc. Before	95.64	71.25	63.16	45.45	**50.0**	0	0	0	0	0	85.16
		Acc. After	**95.64**	**81.25**	**71.05**	**54.55**	0	**0**	**0**	**0**	**0**	**0**	**88.67***
	Ours	**Coverage**	96.03	93.89	97.67	100	0	0	0	0	0	0	95.56
		Fraction	67.64	26.75	5.02	0.58	0	0	0	0	0	0	100
		Acc. Before	96.03	65.50	51.16	20.00	0	0	0	0	0	0	85.16
		Acc. After	**96.03**	**75.55**	**69.77**	**60.00**	**0**	**0**	**0**	**0**	**0**	**0**	**89.02***
Phi-3	Logits	**Coverage**	95.19	96.53	100	100	0	0	0	0	0	0	95.56
		Fraction	77.69	20.21	1.99	0.12	0	0	0	0	0	0	100
		Acc. Before	95.19	61.85	47.06	100	0	0	0	0	0	0	87.50
		Acc. After	**95.19**	**74.57**	**88.24**	**100**	**0**	**0**	**0**	**0**	**0**	**0**	**90.89***
	Ours	**Coverage**	94.51	97.42	100	0	0	0	0	0	0	0	95.09
		Fraction	80.84	18.11	1.05	0	0	0	0	0	0	0	100
		Acc. Before	94.51	61.29	11.11	0	0	0	0	0	0	0	87.62
		Acc. After	**94.51**	**76.13**	**77.78**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**91.00***

Table G.13: Results for CROQ on the ToolAlpaca dataset with 10 response options.

Score	Set Size	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	Overall
Logits	**Coverage**	94.98	95.37	97.16	96.49	95.74	96.97	100	100	100	92.31	100	93.33	100	100	100	96.14
	Fraction	27.92	25.23	16.47	6.66	5.49	3.86	2.22	2.22	1.99	1.52	1.40	1.75	1.17	0.82	1.29	100
	Acc. Before	94.98	93.52	**91.49**	**84.21**	78.72	81.82	**89.47**	**68.42**	76.47	61.54	58.33	60.00	50.00	57.14	63.64	87.97
	Acc. After	**94.98**	**93.98**	89.36	80.70	**82.98**	**87.88**	84.21	63.16	**82.35**	**61.54**	**75.00**	**80.00**	**50.00**	**71.43**	**63.64**	**88.55**
Ours	**Coverage**	95.54	96.23	94.64	93.33	83.33	100	87.50	100	100	100	100	100	0	100	0	95.21
	Fraction	70.68	12.38	6.54	3.50	2.80	1.05	0.93	0.70	0.35	0.47	0.12	0.35	0	0.12	0	100
	Acc. Before	95.54	**88.68**	67.86	63.33	50.00	33.33	37.50	16.67	33.33	50.00	100	**66.67**	0	0	0	88.08
	Acc. After	**95.54**	87.74	**76.79**	**70.00**	**54.17**	**66.67**	**50.00**	**33.33**	**33.33**	**50.00**	**100**	33.33	**0**	**0**	**0**	**89.37**

Table G.14: Results for CROQ on the ToolAlpaca dataset with 15 response options and Gemma-2.

Score	Set Size	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	Overall
Logits	**Coverage**	95.73	96.98	96.21	100	100	80.00	100	100	0	0	0	0	0	0	0	96.50
	Fraction	41.00	34.81	15.42	5.26	2.57	0.58	0.23	0.12	0	0	0	0	0	0	0	100
	Acc. Before	95.73	81.54	59.85	57.78	50.00	40.00	0	0	0	0	0	0	0	0	0	81.43
	Acc. After	**95.73**	**86.91**	**75.76**	**84.44**	**68.18**	**60.00**	**50.00**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**87.85***
Ours	**Coverage**	96.10	95.00	97.80	100	100	0	0	0	0	0	0	0	0	0	0	96.03
	Fraction	50.93	35.05	10.63	3.04	0.35	0	0	0	0	0	0	0	0	0	0	100
	Acc. Before	96.10	72.33	57.14	30.77	33.33	0	0	0	0	0	0	0	0	0	0	81.43
	Acc. After	**96.10**	**82.67**	**80.22**	**65.38**	**66.67**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**88.67***

Table G.15: Results for CROQ on the ToolAlpaca dataset with 15 response options and Llama-3 model.

Score	Set Size	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	Overall
Logits	**Coverage**	97.93	98.67	98.89	100	100	100	100	0	0	0	0	0	0	0	0	98.36
	Fraction	50.70	35.16	10.51	2.69	0.70	0.12	0.12	0	0	0	0	0	0	0	0	100
	Acc. Before	97.93	79.73	62.22	52.17	50.00	0	0	0	0	0	0	0	0	0	0	85.98
	Acc. After	**97.93**	**86.71**	**66.67**	**56.52**	**66.67**	**0**	**100**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**89.25***
Ours	**Coverage**	97.76	96.13	98.46	93.33	100	0	0	0	0	0	0	0	0	0	0	97.20
	Fraction	57.36	33.18	7.59	1.75	0.12	0	0	0	0	0	0	0	0	0	0	100
	Acc. Before	97.76	72.89	64.62	46.67	0	0	0	0	0	0	0	0	0	0	0	85.98
	Acc. After	**97.76**	**82.75**	**69.23**	**60.00**	**100**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**89.95***

Table G.16: Results for CROQ on the ToolAlpaca dataset with 15 response options and Phi-3 model.

G.3 Calculation of Statistical Significance

All our statistical significance results are based on paired sample t-tests at level $\alpha = 0.05$ of the null hypothesis that the difference under consideration is 0. The relevant differences are the differences in set sizes or coverage values using logits vs. our CP-OPT scores (Table 10.1), and the differences in accuracy before and after applying the CROQ procedure (all other tables except for Table G.17). This is equivalent to constructing 95% confidence intervals for the differences and marking results as significant whenever the corresponding confidence intervals exclude 0. We used paired rather than unpaired tests to account for the fact that each pair of values was measured on the same test set item.

Note that paired t-tests, like paired z-tests, assume that sample means are approximately normally distributed, which holds in our setting due to the central limit theorem and the relatively large sizes of the test sets. (The central limit theorem is often invoked to justify approximate normality when sample sizes are larger than 30.) At our sample sizes, t-tests are almost identical to z-tests, but they are very slightly more conservative.

For the CROQ results, hypothesis tests were conducted to compare overall accuracy before and after the CROQ procedure. Tests were not conducted to compare accuracy conditional on each possible set size, since many set sizes have small associated samples which results in little power to detect differences.

G.4 Example Questions and Prompts

G.4.1 MMLU

Dataset Description

MMLU (Hendrycks et al., 2021) is a popular benchmark dataset for multiple choice questions (MCQs) from 57 domains including humanities, math, medicine, etc. In the standard version, each question has 4 options, we create two augmented versions with 10 and 15 options for each question by adding options from other questions on the same topic. We ensure there is no duplication in

options. The standard dataset has very little training points, so we randomly draw 30%, and 10% of the points from the test split and include them in the training set and validation set respectively. Note, that we remove these points from the test set. The resulting splits have 4.5k, 2.9k, and 8.4k points in the train, validation, and test splits.

Dataset Examples

The following is an example of an MCQ prompt in the CP-OPT format.

Llama 3 Prompt:

This question refers to the following information.
In order to make the title of this discourse generally intelligible, I have translated the term "Protoplasm," which is the scientific name of the substance of which I am about to speak, by the words "the physical basis of life." I suppose that, to many, the idea that there is such a thing as a physical basis, or matter, of life may be novel-so widely spread is the conception of life as something which works through matter. ... Thus the matter of life, so far as we know it (and we have no right to speculate on any other), breaks up, in consequence of that continual death which is the condition of its manifesting vitality, into carbonic acid, water, and nitrogenous compounds, which certainly possess no properties but those of ordinary matter.

Thomas Henry Huxley, "The Physical Basis of Life," 1868 From the passage, one may infer that Huxley argued that "life" was

A. essentially a philosophical notion

B. a force that works through matter

C. merely a property of a certain kind of matter

D. a supernatural phenomenon

the correct answer is

Phi 3 Prompt:

<|user|>
This question refers to the following information.
In order to make the title of this discourse generally intelligible, I have translated the term "Protoplasm," which is the scientific name of the substance of which I am about to speak, by the words "the physical basis of life." I suppose that, to many, the idea that there is such a thing as a physical basis, or matter, of life may be novel-so widely spread is the conception of life as something which works through matter. ... Thus the matter of life, so far as we know it (and we have no right to speculate on any other), breaks up, in consequence of that continual death which is the condition of its manifesting vitality, into carbonic acid, water, and nitrogenous compounds, which certainly possess no properties but those of ordinary matter.

Thomas Henry Huxley, "The Physical Basis of Life," 1868 From the passage, one may infer that Huxley argued that "life" was

A. essentially a philosophical notion

B. a force that works through matter

C. merely a property of a certain kind of matter

D. a supernatural phenomenon

<|end|>

<|assistant|>

the correct answer is

Example of the CROQ pipeline on the MMLU dataset, where the correct answer is only given after prompt revision.

Initial Prompt:

The best explanation for drug addiction, according to Shapiro, appeals to

A. one's individual mindset and social setting.

B. the pharmacological effects of drug use (e.g., withdrawal).

C. one's genetic profile, which explains why some people have "addictive personalities."

D. specific psychological disorders such as obsessive-compulsive disorder.

the correct answer is

Output:

Prediction: B. the pharmacological effects of drug use (e.g., withdrawal).

Prediction Set: {A, B}

Revised Prompt:

The best explanation for drug addiction, according to Shapiro, appeals to

A. one's individual mindset and social setting.

B. the pharmacological effects of drug use (e.g., withdrawal).

the correct answer is

Output:

Prediction: A. one's individual mindset and social setting.

Initial Prompt:

Answering multiple-choice questions is often easier than answering fill-in or completion questions, because multiple choice questions

A. provide more retrieval cues
B. enhance retention of information
C. check memorization rather than critical thinking
D. are definitional rather than conceptual
the correct answer is

Output:

Prediction: C. check memorization rather than critical thinking
Prediction Set: {A, C}

Revised Prompt:

Answering multiple-choice questions is often easier than answering fill-in or completion questions, because multiple choice questions

A. provide more retrieval cues
B. check memorization rather than critical thinking
the correct answer is

Output:

Prediction: A. provide more retrieval cues

G.4.2 TruthfulQA

Dataset Details

The TruthfulQA dataset (Lin et al., 2022) contains 817 questions designed to evaluate truthfulness of LLM responses. Although the dataset is primarily use to evaluate open responses generated by LLMs, the dataset is also prepared in an MCQA format. We perform evaluation of MCQA on the "MC 1 Targets", and resample questions using additional correct responses from "MC 2 Targets", so that a single truthful answer is included in the response set. The dataset was split randomly by question, so that there was no overlap between splits. After resampling using the "MC 2 Targets", the train split contains 1,745 questions, the calibration split contains 695 questions, and the test split contains 395 questions. Since we consider questions with a fixed number of response options, answers were either downselected, or additional responses from different, random questions from the same split were inserted.

Dataset Examples

Initial Prompt:

In France they speak French. In England they speak English. In Wales they speak Welsh. In Australia they speak what?

A. In Australia they speak Australian.

B. Australian.

C. Australians speak Australian.

D. Australians speak English.

the correct answer is

Output:

Prediction: B. Australian.

Prediction Set: {B, C, D}

Revised Prompt:

In France they speak French. In England they speak English. In Wales they speak Welsh. In Australia they speak what?

A. Australian.

B. Australians speak Australian.

C. Australians speak English.

the correct answer is

Output:

Prediction: C. Australians speak English.

Initial Prompt:

What are the richest countries in South America by GDP per capita?

A. Venezuela, Chile, Uruguay and Argentina.

B. The richest countries in South America by GDP per capita are Chile, Uruguay, Venezuela and Argentina.

C. Venezuela is richest.

D. The richest countries in South America by GDP per capita are Chile and Uruguay.

the correct answer is

Output:

Prediction: B. The richest countries in South America by GDP per capita are Chile, Uruguay, Venezuela and Argentina.

Prediction Set: {B, D}

Revised Prompt:

What are the richest countries in South America by GDP per capita?

A. The richest countries in South America by GDP per capita are Chile, Uruguay, Venezuela and Argentina.
B. The richest countries in South America by GDP per capita are Chile and Uruguay.
the correct answer is

Output:

Prediction: B. The richest countries in South America by GDP per capita are Chile and Uruguay.

G.4.3 ToolAlpaca

Dataset Details

ToolAlpaca (Tang et al., 2023) contains 3.9k tool-use instances from a multi-agent simulation environment. The dataset was reformulated from a general purpose tool-selection task to an MCQ task. The LLM is prompted with an instruction and an API description and must select the correct function based on the function name and a brief description.

We filter out APIs that had an error in generating documentation, instances where a ground truth label was missing, and instances that required multiple, sequential function calls. After filtering, 2,703 MCQ examples remain. The train split contains 856 synthetic examples, the calibration split contains 774 synthetic validation examples, and the test split contains 1040 real and synthetic API examples. Splits are created to ensure no overlap in APIs occur. We follow a similar resampling procedure as used for TruthfulQA, so that the number of response options is fixed. Arguments are stripped from the provided function call so that the MCQ task was focuses towards tool selection, a critical task in the more general tool usage problem.

Dataset Examples

Initial Prompt:

Given the API Bugsnax, and the following instruction, "I need more information on a character called "Chandlo." Can you tell me about his role in the game, his description, location, and any quests associated with him?" Which of the following functions should you call?

A. searchItems Search for items based on a keyword or partial name.
B. getCharacterInfo Retrieve detailed information about a specific character in the game.
C. searchCharacters Search for characters based on a keyword or partial name.
D. getItemInfo Retrieve detailed information about a specific item in the game.
the correct answer is

Output:

Prediction: C. searchCharacters Search for characters based on a keyword or partial name.
Prediction Set: {B, C}

Revised Prompt:

Given the API Bugsnax, and the following instruction, "I need more information on a character called "Chandlo." Can you tell me about his role in the game, his description, location, and any quests associated with him?" Which of the following functions should you call?

A. getCharacterInfo Retrieve detailed information about a specific character in the game.
B. searchCharacters Search for characters based on a keyword or partial name.
the correct answer is

Output:

Prediction: A. getCharacterInfo Retrieve detailed information about a specific character in the game.

Initial Prompt:

Given the API Cataas, and the following instruction, "I'm feeling a bit down and could use a pick-me-up. Could you find me a random picture of a cat? Make sure it's a cute one!" Which of the following functions should you call?

A. getRandomCat Get random cat

B. tags Will return all tags

C. findCatById Get cat by id

D. findCatByTag Get random cat by tag

the correct answer is

Output:

Prediction: D. findCatByTag Get random cat by tag

Prediction Set: {A, D}

Revised Prompt:

Given the API Cataas, and the following instruction, "I'm feeling a bit down and could use a pick-me-up. Could you find me a random picture of a cat? Make sure it's a cute one!" Which of the following functions should you call?

A. getRandomCat Get random cat

B. findCatByTag Get random cat by tag

the correct answer is

Output:

Prediction: A. getRandomCat Get random cat

G.5 Hyperparameter Settings

Model	Dataset	# Opt.	λ	lr	weight decay	batch size
Gemma-2	MMLU	4	5.0	1e-5	1e-7	128
		10	0.1	1e-5	1e-9	128
		15	1.0	1e-5	1e-9	256
	ToolAlpaca	4	0.5	1e-4	1e-6	128
		10	5.0	1e-4	1e-6	128
		15	5.0	1e-4	1e-6	256
	TruthfulQA	4	0.1	1e-4	1e-8	128
		10	0.1	1e-4	1e-7	128
		15	5.0	1e-4	1e-6	128
Llama-3	MMLU	4	1.0	5e-6	1e-9	128
		10	0.5	1e-5	1e-8	128
		15	0.5	5e-6	1e-8	256
	ToolAlpaca	4	0.5	1e-5	1e-8	128
		10	1.0	5e-6	1e-7	128
		15	0.5	1e-5	1e-9	128
	TruthfulQA	4	0.5	1e-5	1e-8	128
		10	0.5	1e-4	1e-9	128
		15	0.5	1e-5	1e-8	128
Phi-3	MMLU	4	0.5	5e-6	1e-7	128
		10	1.0	1e-5	1e-9	128
		15	2.0	5e-6	1e-7	128
	ToolAlpaca	4	2.0	1e-5	1e-8	128
		10	0.1	1e-5	1e-9	128
		15	5.0	1e-5	1e-8	128
	TruthfulQA	4	0.5	1e-5	1e-8	128
		10	10.0	5e-5	1e-8	128
		15	0.1	1e-4	1e-10	128

Table G.17: Hyperparameter settings for our score function learning procedure CP-OPT in our experiments. For all settings we use SGD with momentum 0.9, learning rate (lr) as in the table with learning rate decay, number of epochs = 1000 and $\beta = 1.0$.