ROBUST DEEP LEARNING UNDER DISTRIBUTION SHIFT

by

Jiefeng Chen

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

Doctor of Philosophy

(Computer Sciences)

at the

UNIVERSITY OF WISCONSIN–MADISON

2023

Date of final oral examination: 06/19/2023

The dissertation is approved by the following members of the Final Oral Committee:
Somesh Jha, Professor, Computer Sciences
Yingyu Liang, Assistant Professor, Computer Sciences
Yong Jae Lee, Associate Professor, Computer Sciences
Kassem Fawaz, Assistant Professor, Electrical & Computer Engineering

For my parents and sisters.

ACKNOWLEDGMENTS

I am filled with profound gratitude towards my parents and sisters for their unwavering support throughout my journey. Their boundless love, unwavering encouragement, and profound understanding have been the steadfast pillars that have sustained me during the most challenging moments. I consider myself truly fortunate to have such exceptional individuals in my life, whose presence has made all the difference.

I am immensely grateful to my Ph.D. advisors, Yingyu Liang and Somesh Jha, for their invaluable guidance, expertise, and continuous encouragement. Their profound knowledge, dedication, and passion for research have inspired me to push the boundaries of my capabilities and strive for excellence. Their mentorship has been instrumental in shaping my research trajectory and personal growth. Thanks to their guidance, I have published several papers in top-tier machine learning conferences and developed into a proficient researcher. Furthermore, their support enabled me to pursue internships at Google, providing invaluable industry experience.

I would also like to wholeheartedly express my appreciation to my esteemed thesis committee members, Yong Jae Lee and Kassem Fawaz, for their insightful feedback, constructive criticism, and invaluable suggestions. I am deeply grateful for the time and effort they dedicated to meticulously reviewing and evaluating my research, which has significantly shaped its overall excellence. Their expertise and willingness to share their vast knowledge have been instrumental in my academic growth, and I am truly thankful for their unwavering support throughout the thesis process.

Furthermore, I am compelled to acknowledge and extend my deepest gratitude to all the collaborators who have played an integral role in shaping my Ph.D. journey. Their invaluable contributions, collaborations, and fruitful discussions have not only enriched my research experience but also expanded the horizons of my work. Among them, I reserve a special place of appreciation for Xi Wu, whose unwavering support and invaluable assistance during the formative years of my Ph.D. journey

have played a pivotal role in shaping my growth as a researcher. I am sincerely thankful for the profound impact they have had on my academic pursuits.

Lastly, I am humbled and deeply indebted to the numerous individuals who have supported me in diverse ways throughout my academic pursuit. To my cherished friends and esteemed colleagues, I extend my heartfelt gratitude for the unwavering camaraderie, endless encouragement, and constant source of inspiration you have provided. Your presence has not only made this journey more meaningful and enjoyable but has also played an instrumental role in shaping my academic and personal growth.

In conclusion, I am profoundly grateful to all those who contributed to my Ph.D. journey, directly or indirectly. Without their support, this achievement would not have been possible.

CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABSTRACT

Deep learning has achieved remarkable success in various domains, including computer vision, natural language processing, and game playing. However, this success relies on the assumption that the distribution of the test data is identical to the distribution of the training data. In practice, this assumption often does not hold, leading to distribution shifts. There are two types of distribution shifts that occur in real-world scenarios: naturally occurring shifts during data collection and shifts intentionally crafted by adversaries to undermine machine learning systems. It has been observed that deep neural networks (DNNs) typically experience a significant performance decline under distribution shifts.

To address this issue, my research focuses on three main directions. Firstly, we investigate robust deep learning techniques aimed at enhancing the adversarial robustness of DNNs. This involves proposing and formulating novel defenses, such as transductive-learning based defenses and defenses based on rejection. We also develop robust attack frameworks to evaluate the effectiveness of these defenses. Additionally, we explore robustness beyond prediction, examining aspects such as robust attribution and robust out-of-distribution detection.

Secondly, we tackle the challenge of reliable model deployment by proposing a new framework to estimate the generalization capabilities of DNNs during testing. We also introduce an innovative learning paradigm called active selective prediction to achieve better utilization of humans in the loop for reliable model deployment.

Lastly, we delve into foundation models and provide new insights into their training. Our findings reveal a trade-off between the universality and label efficiency of model representations trained through contrastive learning. Furthermore, we present a novel framework for adaptation with self-evaluation, aiming to improve the selective prediction performance of Large Language Models (LLMs) and ensure their reliable deployment in real-world applications.

By addressing the challenges posed by distribution shifts in these three research directions, our work contributes to the development of more robust deep learning systems capable of handling real-world scenarios effectively.

1 INTRODUCTION

Deep learning has achieved remarkable success in various domains, including computer vision (He et al., 2016a; Ren et al., 2015; Chen et al., 2018; Goodfellow et al., 2014; Krizhevsky et al., 2012), natural language processing (Devlin et al., 2019; Vaswani et al., 2017; Williams et al., 2018; Dolan and Brockett, 2005; Howard and Ruder, 2018; Rajpurkar et al., 2016; Peters et al., 2018), speech recognition (Hannun et al., 2014; Abdel-Hamid et al., 2014; Amodei et al., 2016; Chorowski et al., 2015) and game playing (Silver et al., 2016, 2017), etc. Despite their success, deep neural networks (DNNs) have been shown to be vulnerable to distribution shifts, limiting their deployment in safety-critical domains (Sharma et al., 2019; Eykholt et al., 2018). Specifically, while DNNs excel when the test data aligns with the training data distribution, they often experience significant performance degradation when faced with distribution shifts. For instance, object detectors trained on ImageNet may exhibit a 40-45% drop in performance when evaluated on ObjectNet (Barbu et al., 2019). This raises a critical research question: *How can we develop deep learning systems that are robust under distribution shifts?*

In this thesis, we assert the following statement:

> *Developing novel techniques and methodologies to enhance the robustness of deep learning systems in the face of distribution shifts is crucial for enabling their reliable and effective deployment in real-world applications.*

In the real world, distribution shifts leading to performance degradation primarily fall into two categories: (1) adversarial attacks (Papernot et al., 2016), where imperceptible perturbations are added to test inputs to induce incorrect predictions; and (2) natural distribution shifts (Quiñonero-Candela et al., 2008), typically resulting from changes in the image generation pipeline (Torralba and Efros, 2011) or noise corruptions (Hendrycks and Dietterich, 2019). Adversarial attacks aim to induce model failure by introducing a bounded worst-case distribution shift, whereas natural distribution shifts occur without any adversarial intervention. Examples of natural distribution shifts include covariate shift due to domain divergence, image

corruptions due to weather conditions, and out-of-distribution (OOD) test inputs originating from open-world environments. Adversarial attacks are commonly applied to clean test inputs sampled from the training data distribution, but they can also be employed on test inputs with natural distribution shifts to create even stronger shifts.

This thesis tackles the challenge posed by these two categories of distribution shifts through contributions in three key directions. The first direction is robust deep learning, which aims to enhance the resilience of DNNs against adversarial attacks. The objective is to develop techniques that effectively strengthen the DNNs' ability to withstand such attacks. The second direction focuses on reliable model deployment, aiming to establish techniques that ensure the safe and effective deployment of DNNs in real-world scenarios. This involves addressing challenges such as unsupervised accuracy estimation, error detection, and active selective prediction. The third direction explores foundation models, which are trained on extensive unlabeled data at scale and can be adapted to diverse downstream tasks, even in the presence of significant distribution shifts. In the subsequent sections, we will delve into the specific details of these three directions and discuss their respective contributions.

1.1 Robust Deep Learning

Adversarial examples (Szegedy et al., 2014; Biggio et al., 2013a) are inputs deliberately crafted by adversaries to induce erroneous outputs from machine learning systems. As the use of machine learning becomes more widespread, there has been significant recent research on developing robust models to defend against adversarial attacks (Dalvi et al., 2004; Barreno et al., 2006; Globerson and Roweis, 2006; Barreno et al., 2010; Biggio et al., 2010; Šrndic and Laskov, 2013). Carlini et al. identify three common motivations for studying defenses against adversarial examples: (1) to defend against an adversary who will attack the system; (2) to test the worst-case robustness of machine learning algorithms; (3) to measure progress of machine learning algorithms towards human-level abilities.

In this thesis, we contribute to enhancing the adversarial robustness of various aspects of deep learning systems, including robust prediction, robust attribution, and robust OOD detection. We provide detailed explanations of these contributions below.

Robust prediction. Despite the numerous defense mechanisms proposed for robust prediction, many of them have been shown to be vulnerable to strong adaptive attacks (Tramèr et al., 2020; Athalye et al., 2018). Adversarial training and its variants have proven to be the most effective methods for learning robust models (Madry et al., 2018; Zhang et al., 2019). However, achieving satisfactory robust accuracy on complex datasets remains a challenge. For example, as reported in RobustBench (Croce et al., 2020), even state-of-the-art adversarially trained models struggle to exceed 70% robust test accuracy on CIFAR-10. Moreover, the robust models usually have poor generalization to threat models that are not utilized during training (Stutz et al., 2020; Laidlaw et al., 2021). To address these challenges, we investigate two novel defense approaches: transductive-learning based defenses and defenses with a rejection option. We propose a novel attack framework Greedy Model Space Attack (GMSA) that can serve as a strong baseline for evaluating transductive-learning based defenses (Chen et al., 2021c). Through systematic evaluation, we demonstrate that GMSA can break previous transductive-learning based defenses and adversarially retraining the model using fresh randomness at the test time gives a significant increase in robustness against the proposed GMSA attack. Additionally, we study adversarially-robust classification with rejection in the stratified rejection setting, which has rejection loss functions that are monotonically non-increasing in the perturbation magnitude. We propose new evaluation metrics, namely the *total robust loss* and the *robustness curve*, that provide a more fine-grained evaluation of adversarial robustness with rejection. We theoretically analyze the stratified rejection setting and propose a novel defense method – *Adversarial Training with Consistent Prediction-based Rejection* (CPR) – for building a robust selective classifier (Chen et al., 2023a). Extensive experiments demonstrate that the proposed method CPR significantly outperforms previous methods under strong

adaptive attacks.

Robust attribution. As Ghorbani et al. convincingly demonstrated, for existing DNNs, one can generate minimal perturbations that substantially change the DNNs' interpretations, while keeping their correct predictions intact. Thus, images that look very similar to a human may have very different attributions, a phenomenon that can seriously erode trust in such models. For example, when a doctor uses a deep model as a diagnostic tool, they usually want to know which features cause the model to detect a certain disease. If the attributions are brittle, doctors will find it difficult to trust the model. Therefore, it is important to train models that produce reliable interpretations for their predictions. We take a step towards solving this problem by viewing it through the lens of axiomatic attribution of neural networks, and propose Robust Attribution Regularization (RAR) (Chen et al., 2019). RAR aims to regularize the training so the resulting model will have robust attributions that are not substantially changed under minimal input perturbations. Our theoretical analysis reveals that RAR gives principled generalizations of previous objectives designed for robust predictions. Our empirical evaluation demonstrates that RAR gives significantly better attribution robustness and also gives comparable prediction robustness (sometimes even better), compared to adversarial training.

Robust OOD detection. Prior research has demonstrated the generation of adversarial OOD examples by slightly perturbing clean OOD examples, causing OOD detectors to misclassify them as in-distribution (ID) inputs (Sehwag et al., 2019). Failing to detect adversarial OOD examples can have severe consequences in safety-critical applications like autonomous driving. Motivated by this, we make an important step towards the robust OOD detection problem, and propose a novel training framework, ***A**dversarial **T**raining with informative **O**utlier **M**ining* (ATOM) (Chen et al., 2021a). We extensively evaluate ATOM on common OOD detection benchmarks, as well as a suite of adversarial OOD tasks, and demonstrate that ATOM achieves state-of-the-art performance under a broad family of classic and adversarial OOD evaluation tasks.

With our contributions in enhancing robust prediction, robust attribution, and robust OOD detection, we aim to advance the field of adversarial machine learning and promote the development of more reliable and secure deep learning systems.

1.2 Reliable Model Deployment

DNNs typically suffer a significant performance reduction when faced with distribution shifts. Consequently, post-deployment monitoring of trained model performance is crucial to ensure safe deployment. This monitoring encompasses two challenging tasks: unsupervised accuracy estimation, which aims to estimate the accuracy of the model on unlabeled test data, and error detection, which aims to identify individual misclassified inputs so that we know where to improve the model. Our research contributes to these challenging aspects by introducing a novel algorithmic framework – *Self-Training Ensembles* – which tackles both unsupervised accuracy estimation and error detection simultaneously (Chen et al., 2021b). We provide theoretical analysis identifying the success conditions of the framework and perform extensive experiments to demonstrate that the instantiations of the framework achieve state-of-the-art performance on both tasks.

When a performance drop is detected in the deployed model due to distribution shifts in the test data, proactive measures must be taken to prevent performance degradation. One effective solution involves incorporating humans in the loop. Selective prediction serves as a means to involve humans in the loop: DNNs can abstain from making predictions when model uncertainty is high, and these uncertain predictions can be deferred to human experts for further evaluation. However, distribution shifts can lead to more inaccurate predictions, which necessitates increased human labeling in selective prediction. Unfortunately, this can be challenging and costly in many scenarios. Active learning mitigates this difficulty by selectively querying the most informative examples, reducing overall labeling effort in several cases.

Based on this motivation, this thesis presents a novel learning paradigm called *active selective prediction*, which learns to query more informative samples from

the shifted target domain while increasing accuracy and coverage in selective prediction. To solve the active selective prediction problem, we propose a simple but effective method, ASPEST, that trains ensembles of model snapshots using self-training with their aggregated outputs as pseudo labels (Chen et al., 2023b). Extensive experiments demonstrate that the proposed method ASPEST significantly outperforms previous methods in selective prediction and active learning, leading to more optimal utilization of humans in the loop.

1.3 Foundation Models

In the traditional approach, DNNs are trained using a substantial amount of labeled data within a supervised learning framework, assuming that both the training and test data follow the same underlying distribution. However, obtaining a large labeled dataset is often a difficult and expensive task in real-world situations. Moreover, practical scenarios often involve test data distributions that differ from those of the training data. As a consequence of limited human-annotated data and the presence of distribution shifts, DNNs tend to exhibit subpar performance when tested on unseen data.

To overcome these challenges, a new learning paradigm has been developed: we first pre-train a model (e.g., a representation function) on broad unlabeled data at scale and then adapt (e.g., fine-tune) it to a wide range of downstream tasks. It is sometimes referred to as the foundation models (Bommasani et al., 2021). Current examples include BERT (Devlin et al., 2019), GPT-3 (Brown et al., 2020) and CLIP (Radford et al., 2021). From a technological point of view, foundation models are not new – they are based on deep neural networks and self-supervised learning, both of which have existed for decades. Self-supervised learning is a type of unsupervised learning where a model is trained on labels that are automatically derived from the data itself without human annotation. Self-supervised learning methods enable a model to learn useful knowledge about an unlabeled dataset by learning useful representations. It has been shown that this learning paradigm can significantly reduce the number of labeled data needed to build a good classifier.

For example, Chen et al. demonstrated that on ImageNet, with only 10% of labels, ResNet-50 trained with their method under this paradigm achieves 77.5% top-1 accuracy, outperforming standard supervised training with all of the labels. Besides, as demonstrated in Albuquerque et al. (2020), self-supervised pre-training on multiple tasks can learn representations that are robust to test data with distribution shifts.

There are two key desiderata for the foundation models: (1) *label efficiency*: the ability to learn an accurate classifier on top of the representation with a small amount of labeled data; (2) *universality*: the pre-trained representation can be used for various downstream tasks. Due to the universality property, the foundation models can be used for different kinds of distributions and thus can help address the issue of distribution shifts. We study one of the most popular instantiations of this paradigm: contrastive learning with linear probing, i.e., learning a linear predictor on the representation pre-trained by contrastive learning. We show that there exists a trade-off between the two desiderata so that one may not be able to achieve both simultaneously. Specifically, we provide analysis using a theoretical data model and show that, while more diverse pre-training data result in more diverse features for different tasks (improving universality), it puts less emphasis on task-specific features, giving rise to larger sample complexity for down-stream supervised tasks, and thus worse prediction performance. Guided by this analysis, we propose a contrastive regularization method to improve the trade-off (Shi et al., 2022a). We validate our analysis and method empirically with systematic experiments using real-world datasets and foundation models.

Large Language Models (LLMs) belong to the family of foundation models and have recently made remarkable progress in various tasks, such as natural language understanding and generation. However, their use in high-stakes decision-making scenarios is still limited due to the potential for errors. Selective prediction is a technique that can be used to improve the reliability of the LLMs by allowing them to abstain from making predictions when they are unsure of the answer. In this thesis, we propose a novel framework ASPIRE for adaptation with self-evaluation to improve the selective prediction performance of LLMs. Our framework is based

on the idea of using parameter-efficient tuning to adapt the LLM to the specific task at hand while improving its ability to perform self-evaluation. We evaluate our method on a variety of question-answering (QA) datasets and show that it outperforms state-of-the-art selective prediction methods.

In summary, this thesis makes significant contributions to the field of foundation models. Firstly, it uncovers the trade-off between universality and label efficiency of representations from contrastive learning. By investigating this trade-off, we gain a deeper understanding of the factors influencing the effectiveness of these representations. Secondly, it proposes a novel framework called ASPIRE, which focuses on adaptation with self-evaluation to improve the selective prediction performance of LLMs. Through the development of ASPIRE, we offer a practical solution that enhances the reliability and performance of LLMs in high-stakes decision-making scenarios. Together, these contributions advance the field of foundation models and pave the way for further advancements in representation learning and selective prediction techniques.

1.4 Contributions and Remaining Challenges

The thesis statement emphasizes the importance of developing deep learning systems that are robust under distribution shifts and proposes the need for novel techniques and methodologies for their reliable and effective deployment in real-world applications. We have presented the contributions made in three key directions: robust deep learning, reliable model deployment, and foundation models. In robust deep learning, we have developed new frameworks to enhance the performance of machine learning systems in robust prediction, robust attribution, and robust OOD detection tasks. For reliable model deployment, we have proposed frameworks to estimate the generalization of DNNs during test time, along with introducing a new learning paradigm called active selective prediction. In the field of foundation models, we have made notable progress in understanding the trade-off between universality and label efficiency. Additionally, we have proposed a novel framework for adaptation with self-evaluation to improve the selective prediction performance

of LLMs. These contributions provide substantial progress in achieving the thesis statement.

However, despite these significant contributions, there are remaining challenges that must be addressed to fully achieve the vision of developing robust deep learning systems under distribution shifts. One major challenge is the ongoing development of more effective defense mechanisms against adversarial attacks. Adversarial attacks constantly evolve, and it is essential to stay ahead of the evolving landscape of attack strategies. This requires continuous research and innovation to develop robust defense mechanisms that can withstand increasingly sophisticated attacks.

Another challenge that needs further exploration is the scalability and generalization of the proposed methodologies. While our research has demonstrated promising results, it is essential to investigate the scalability of the developed techniques to larger and more complex datasets. Additionally, the generalization of these methodologies to different domains and applications is an important area of future research. Ensuring that the developed techniques can be applied effectively in diverse real-world scenarios will significantly enhance the practicality and impact of our work.

Furthermore, the trade-off between universality and label efficiency in foundation models is an ongoing challenge. Our research has shed light on this trade-off and provided insights into its implications. However, further refinement and optimization of this trade-off are necessary to strike the right balance between the universality of pre-trained representations and the sample efficiency of downstream supervised adaptation tasks. By addressing this challenge, we can unlock the full potential of foundation models in addressing distribution shifts and improving the performance of deep learning systems in various domains.

In conclusion, while the presented contributions have made significant progress in accomplishing the thesis statement, there are remaining challenges that need to be tackled to fully realize the vision of developing robust deep learning systems under distribution shifts. By addressing these challenges, we can continue to advance the field and enable the reliable and effective deployment of deep learning systems in real-world applications.

1.5 Thesis Outline

This thesis is organized as follows:

In chapters 2 - 5, we present our contributions to robust deep learning. Chapter 2 focuses on transductive adversarial robustness, where we introduce the principle of attacking model space and propose GMSA for evaluating the adversarial robustness of transductive-learning based defenses (Chen et al., 2021c). Moving to Chapter 3, we delve into adversarially-robust classification with rejection in the stratified rejection setting. We theoretically analyze the stratified rejection setting and propose a novel defense method CPR for building a robust selective classifier (Chen et al., 2023a). Chapter 4 centers around attribution robustness, where we propose a novel method RAR that can significantly improve the attribution robustness (Chen et al., 2019). In Chapter 5, we investigate robust OOD detection. Here, we present a novel training framework called ATOM that can significantly improve the robustness of an OOD detector and generalize to unseen adversarial attacks (Chen et al., 2021a).

Chapters 6 - 7 detail our contributions to reliable model deployment. In Chapter 6, we study two challenging tasks – unsupervised accuracy estimation and error detection. We propose a novel framework called self-training ensembles to solve these two challenging tasks simultaneously (Chen et al., 2021b). Chapter 7 introduces a new problem called active selective prediction, for which we propose a simple yet effective solution ASPEST (Chen et al., 2023b).

Chapters 8 - 9 cover our contributions to foundation models. Chapter 8 investigates the trade-off between universality and label efficiency of representations in contrastive learning. We provide empirical evidence for the existence of this trade-off and analyze it theoretically to gain a deeper understanding of why it occurs (Shi et al., 2022a). Moving to Chapter 9, we study selective prediction for LLMs. We propose a novel framework ASPIRE for adaptation with self-evaluation to improve the selective prediction performance of LLMs.

Finally, in Chapter 10, we conclude the thesis and discuss future research directions.

2 TOWARD EVALUATING THE ROBUSTNESS OF NEURAL NETWORKS LEARNED BY TRANSDUCTION

Contribution statement. This chapter is joint work with Xi Wu, Yang Guo, Yingyu Liang, and Somesh Jha. The author Jiefeng Chen proposed the method and completed all the experiments. The paper version of this chapter appeared in ICLR 2022 (Chen et al., 2021c).

2.1 Introduction

Adversarial robustness of deep learning models has received significant attention in recent years (see Kolter and Madry (2018) and references therein). The classic threat model of adversarial robustness considers an *inductive setting* where a model is learned at the training time and fixed, and then at the test time, an attacker attempts to thwart the fixed model with adversarially perturbed input. This gives rise to the adversarial training (Madry et al., 2018; Sinha et al., 2018; Schmidt et al., 2018; Carmon et al., 2019) to enhance adversarial robustness.

Going beyond the inductive threat model, there has been emerging interest in using *transductive learning* (Vapnik, 1998)[1] for adversarial robustness (Goldwasser et al., 2020; Wu et al., 2020b; Wang et al., 2021a). In essence, these defenses attempt to leverage *a batch* of *test-time inputs*, which is common for ML pipelines deployed with batch predictions (bat, 2021), to *learn an updated model*. The hope is that this "test-time learning" may be useful for adversarial robustness since the defender can adapt the model to the perturbed input from the adversary, which is distinct from the inductive threat model where a model is fixed after training.

This chapter examines these defenses from a principled threat analysis perspective. We first formulate and analyze rigorous threat models. Our basic 1-round

[1]We note that this type of defense goes under different names such as "test-time adaptation" or "dynamic defenses". Nevertheless, they all fall into the classic transductive learning paradigm (Vapnik, 1998), which attempts to leverage test data for learning. We thus call them *transductive-learning based defenses*. The word "transductive" is also adopted in Goldwasser et al. (2020).

threat model considers a single-round game between the attacker and the defender. Roughly speaking, the attacker uses an objective $\max_{V' \in N(V)} L_a(\Gamma(U'), V')$ (formula (2.2)), where V is the given test batch, $N(V)$ is a neighborhood around V, L_a is a loss function for attack gain, Γ is the transductive-learning based defense, and $U' = V'|_X$, the projection of V' to features, is the adversarially perturbed data for breaking Γ. This objective is *transductive* as U', the attacker's output, appears in both attack (V' in L_a) and defense (U' in Γ). We extend this threat model to multiple rounds, which is necessary when considering DENT (Wang et al., 2021a) and RMC (Wu et al., 2020b). We point out important subtleties in the modeling that were unclear or overlooked in previous work.

We then study *adaptive attacks*, that is to leverage the knowledge about Γ to construct attacks. Compared to situations considered in BPDA (Athalye et al., 2018), a transductive learner Γ is even further from being differentiable, and theoretically the attack objective is a bilevel optimization (Colson et al., 2007). To address these difficulties, our key observation is to consider the *transferability of adversarial examples*, and consider a *robust* version of (2.2): $\max_{U'} \min_{\overline{U} \in \mathcal{N}(U')} L_a(\Gamma(\overline{U}), V')$ (formula (2.6)), where we want to find a *single* attack set U' to thwart a family of models, induced by $\overline{U}$ "around" U'. This objective relaxes the attacker-defender constraint, and provides more information in dealing with nondifferentiability. To solve the robust objective, we propose Greedy Model Space Attack (GMSA), a general attack framework which attempts to solve the robust objective in a greedy manner. GMSA can serve as a new baseline for evaluating transductive-learning based defenses.

We perform a systematic empirical study on various defenses. For RMC (Wu et al., 2020b), DENT (Wang et al., 2021a), and URejectron (Goldwasser et al., 2020), we show that even weak instantiations of GMSA can break respective defenses. Specifically, for defenses based on adversarially training, we reduce the robust accuracy to that of adversarial training alone. We note that, under AutoAttack (Croce and Hein, 2020), the state-of-the-art adaptive attack for the inductive threat model, some of these defenses have claimed to achieve *substantial improvements* compared to *adversarial training alone*. For example, Wang et al. show that DENT can improve

the robustness of the state-of-the-art adversarial training defenses by more than 20% absolutely against AutoAttack on CIFAR-10. However, under our adaptive attacks, DENT only has minor improvement: less than 3% improvement over adversarial training alone. Our results thus demonstrates significant differences between attacking transductive-learning based defenses and attacking in the inductive setting, and significant difficulties in the use of transductive learning to improve adversarial robustness. On the positive side, we report a somewhat surprising empirical result of *transductive adversarial training*: Adversarially retraining the model using fresh private randomness on a new batch of test-time data gives a significant increase in robustness against all of our considered attacks.

2.2 Preliminaries

Let F be a model, and for a data point $(\mathbf{x}, y) \in \mathcal{X} \times \mathcal{Y}$, a loss function $\ell(F; \mathbf{x}, y)$ gives the loss of F on the point. Let V be a set of labeled data points, and let $L(F, V) = \frac{1}{|V|} \sum_{(\mathbf{x},y)\in V} \ell(F; \mathbf{x}, y)$ denote the empirical loss of F on V. For example, if we use binary loss $\ell^{0,1}(F; \mathbf{x}, y) = \mathbb{1}[F(\mathbf{x}) \neq y]$, this gives the test error of F on V. We use the notation $V|_X$ to denote the projection of V to its features, that is $\{(\mathbf{x}_i, y_i)\}_{i=1}^m|_X \mapsto \{\mathbf{x}_i\}_{i=1}^m$. Throughout this chapter, we use $N(\cdot)$ to denote a neighborhood function for perturbing features: That is, $N(\mathbf{x}) = \{\mathbf{x}' \mid d(\mathbf{x}', \mathbf{x}) < \epsilon\}$ is a set of examples that are close to $\mathbf{x}$ in terms of a distance metric d (e.g., $d(\mathbf{x}', \mathbf{x}) = \|\mathbf{x}' - \mathbf{x}\|_p$). Given $U = \{\mathbf{x}_i\}_{i=1}^m$, let $N(U) = \{\{\mathbf{x}'_i\}_{i=1}^m \mid d(\mathbf{x}'_i, \mathbf{x}_i) < \epsilon, i = 0, \ldots, m\}$. Since labels are not changed for adversarial examples, we also use the notation $N(V)$ to denote perturbations of features, with labels fixed.

2.3 Modeling Transductive Robustness

In this section we formulate and analyze threat models for transductive defenses. We first formulate a threat model for a single-round game between the attacker and the defender. We then consider extensions of this threat model to multiple rounds, which are necessary when considering DENT (Wang et al., 2021a) and

RMC (Wu et al., 2020b), and point out important subtleties in modeling that were not articulated in previous work. We characterize previous test-time defenses using our threat models.

1-round game. In this case, the adversary "intercepts" a clean test data V (with clean features $U = V|_X$, and labels $V|_Y$), adversarially perturbs it, and sends a perturbed features U' to the defender. The defender learns a new model based on U'. A referee then evaluates the accuracy of the adapted model on U'. Formally:

Definition 2.1 (***1-round threat model for transductive adversarial robustness***)**.** *Fix an adversarial perturbation type (e.g., ℓ_∞ perturbations with perturbation budget ε). Let $P_{X,Y}$ be a data generation distribution. The attacker is an algorithm $\mathcal{A}$, and the defender is a pair of algorithms $(\mathcal{T}, \Gamma)$, where $\mathcal{T}$ is a supervised learning algorithm, and Γ is a transductive learning algorithm. A (clean) training set D is sampled i.i.d. from $P_{X,Y}$. A (clean) test set V is sampled i.i.d. from $P_{X,Y}$.*

• **[Training time, defender]** *The defender trains an optional base model $F = \mathcal{T}(D)$, using the labeled source data D.*

• **[Test time, attacker]** *The attacker receives V, and produces an (adversarial) unlabeled dataset U':*

1. *On input Γ, F, D, and V, $\mathcal{A}$ perturbs each point $(\mathbf{x}, y) \in V$ to $(\mathbf{x}', y)$ (subject to the agreed attack type), giving $V' = \mathcal{A}(\Gamma, F, D, V)$ (that is, $V' \in N(V)$).*
2. *Send $U' = V'|_X$ (the feature vectors of V') to the defender.*

• **[Test time, defender]** *The defender produces a model as $F^* = \Gamma(F, D, U')$.*

Multi-round games. The extension of 1-round games to multi-round contains several important considerations that were implicit or unclear in previous work, and is closely related to what it means by *adaptive attacks*. Specifically:

Private randomness. Note that Γ uses randomness, such as random initialization and random restarts[2] in adversarial training. Since these randomness are generated

[2]When perturbing a data point during adversarial training, one starts with a random point in the neighborhood.

after the attacker's move, they are treated as *private randomness*, and *not* known to the adversary.

Intermediate defender states leaking vs. Non-leaking. In a multi-round game, *the defender* may maintain states across rounds. For example, the defender may store test data and updated models from previous rounds, and use them in a new round. If these intermediate defender states are "leaked" to the attacker, we call it *intermediate defender states leaking*, or simply *states leaking*, otherwise we call it *non states-leaking*, or simply non-leaking. Note that the attacker *cannot* simply compute these information by simulating on the training and testing data, due to the use of *private randomness*. We note that, however, the initial pretrained model is assumed to be known by the attacker. The attacker can also of course maintain arbitrary states, and are assumed not known to the defender.

Adaptive vs. Non-adaptive. Because transductive learning happens after the attacker produces U', the attacker may not be able to directly attack the model Γ produced. Nevertheless, the attacker is assumed to have *full knowledge of the transductive mechanism* Γ, except the private randomness. In this chapter, we call an attack *adaptive* if it makes *explicit use of the knowledge of* Γ.

Naturally ordered vs. Adversarially ordered. Both RMC and DENT handle batches of fixed sizes. An intuitive setup for multi-round game is that the batches come in sequentially, and the attacker must forward perturbed versions of these batches *in the same order* to the defender, which we call the "naturally ordered" game. However, this formulation does not capture an important scenario: An adversary can wait and pool a large amount of test data, then chooses a *"worst-case" order* of perturbed data points, and then sends them in batches one at a time for adaptation in order to maximize the breach. We call the latter "adversarially ordered" game. We note that all previous work only considered naturally-ordered game, which gives the defender more advantages, and is thus our focus in the rest of the chapter.

Modeling capacity of our threat models. Our threat models encompass a large family of defenses. For example, without using Γ, the threat model degenerates to the classic inductive threat model. Our threat models also capture various "test-time defenses" proposals (e.g., those broken by the BPDA (Athalye et al., 2018)),

where Γ is a "non-differentiable" function which *"sanitizes" the test data*, instead of updating the model, before sending them to a fixed pretrained model. Therefore, in particular, these proposals are not transductive-learning based. Below we describe previous defenses which we study in the rest of this chapter, where Γ is indeed transductive learning.

Example 2.2 (***Runtime masking and cleansing***)**.** *Runtime masking and cleansing* (*RMC*) (*Wu et al., 2020b*) *is a recent transductive-learning defense. For RMC, the defender is stateful and adapted from the model* learned in the last round*, on a single test point* ($|U| = 1$)*: The adaptation objective is* $F^* = \arg\min_F \sum_{(x,y)\in N'(\widehat{x})} L(F, x, y)$*, where* $\widehat{x}$ *is the test time feature point, and* $N'(\widehat{x})$ *is the set of examples in the adversarial training dataset* D' *that are top-K nearest to* $\widehat{x}$ *in a distance measure. RMC paper considered two attacks:* **(1) *Transfer attack***, *which generates perturbed data by attacking the initial base model, and* **(2) *PGD-skip attack***, *which at round* $p + 1$*, runs PGD attack on the model learned at round* p*. In our language, transfer attack is* stateless (*i.e. the adversary maintains no state*) *and* non-adaptive*, PGD-skip attack is* state-leaking*, but still* non-adaptive*.*

Example 2.3 (***Defensive entropy minimization* (*DENT* (*Wang et al., 2021a*))**)**. *DENT adapts the model using test input, and can work with any training-time learning procedure. The DENT defender is* stateless*: It always starts the adaptation from the original pretrained model, fixed at the training time. During the test-time adaptation, only the affine parameters in batch normalization layers of the base model are updated, using entropy minimization with the information maximization regularization. In this chapter, we show that with strong adaptive attacks under the naturally ordered setting, we are able to reduce the robustness to be almost the same as that of static models. Further, under the adversarially ordered setting, we can completely break DENT. Refer to Section 2.5 for the details.*

Example 2.4 (***Goldwasser et al.'s transductive threat model***)**.** *While seemingly our threat model is quite different from the one described in Goldwasser et al.* (*2020*)*, one can indeed recover their threat model naturally as a* 1-round game*: First, for the perturbation type, we simply allow arbitrary perturbations in the threat model setup. Second, we have a fixed pretrained model* F*, and the adaptation algorithm* Γ *learns a set* S *which represents the*

set of "allowable" points (so $F|_S$ yields a predictor with redaction, namely it outputs $\perp$ for points outside of S). Third, we define two error functions as (5) and (6) in Goldwasser et al. (2020):

$$\underset{U'}{\mathrm{err}}(F|_S, f) \equiv \frac{1}{|U'|}\left|\left\{x' \in U' \cap S \,\middle|\, F(x') \neq f(x')\right\}\right|, \quad \underset{U}{\mathrm{rej}}(S) \equiv \frac{|U \setminus S|}{|U|} \tag{2.1}$$

where f is the ground truth hypothesis. The first equation measures prediction errors in U' that passed through S, and the second equation measures the rejection rate of the clean input. The referee evaluates by measuring two errors: $L(F|_S, V') = \left(\mathrm{err}_{U'}(F|_S), \mathrm{rej}_U(S)\right)$.

2.4 Adaptive Attacks in One Round

In this section we study a basic question: *How to perform adaptive attacks against a transductive-learning based defense in* one round*?* Note that, in each round of a multi-round game, an *independent* batch of test input U is sampled, and the defender can use transductive learning to produce a model specifically adapted to the adversarial input U', *after* the defender receives it. Therefore, it is of fundamental interest to attack this ad-hoc adaptation. We consider white-box attacks: The attacker knows all the details of Γ, except private randomness, which is sampled after the attacker's move.

We deduce a principle for adaptive attacks in one round, which we call *the principle of attacking model space*: Effective attacks against a transductive defense may need to consider *attacking a set of representative models induced in the neighborhood of* U. We give concrete instantiations of this principle, and show in experiments that they break previous transductive-learning based defenses.

Attacks in multi-round. If the transductive-learning based defense is stateless, then we simply repeat one-round attack multiple times. If it is stateful, then we need to consider state-leaking setting or non-leaking setting. For all experiments in Section 2.5, we only evaluate *non-leaking* setting, which is more challenging for the adversary.

2.4.1 Goal of the attacker and challenges

To start with, given a defense mechanism Γ, the objective of the attacker can be formulated as:

$$\max_{V' \in N(V), U' = V'|_X} L_a(\Gamma(F, D, U'), V'). \tag{2.2}$$

where L_a is the loss function of the attacker. We make some notational simplifications: Since D is a constant, in the following we drop it and write $\Gamma(U')$. Also, since the attacker does not modify the labels in the threat model, we abuse the notation and write the objective as

$$\max_{V', U' = V'|_X} L_a(\Gamma(U'), U'). \tag{2.3}$$

A generic attacker would proceed iteratively as follows: It starts with the clean test set V, and generates a sequence of (*hopefully*) *increasingly stronger* attack sets $U^{(0)} = V|_X, U^{(1)}, \ldots, U^{(i)}$ ($U^{(i)}$ must satisfy the attack constraints at U, such as ℓ_∞ bound). We note several basic but important *differences* between transductive attacks and inductive attacks in the classic minimax threat model:

(D1) $\Gamma(U')$ is *not* differentiable. For the scenarios we are interested in, Γ is an optimization algorithm to solve an objective $F^* \in \arg\min_F L_d(F, D, U')$. This renders (2.3) into a bilevel optimization problem (Colson et al., 2007):

$$\max_{V' \in N(V); U' = V'|_X} L_a(F^*, V') \quad \text{subject to: } F^* \in \arg\min_F L_d(F, D, U'), \tag{2.4}$$

In these cases, Γ is in general *not* (in fact far from) differentiable. A natural attempt is to approximate Γ with a differentiable function, using theories such as Neural Tangent Kernels (Jacot et al., 2018). Unfortunately no existing theory applies to the transductive learning, which deals with unlabeled data U' (also, as we have remarked previously, tricks such as BPDA (Athalye et al., 2018) also does not apply because transductive learning is much more complex than test-time defenses considered there).

(D2) U' appears in *both* attack and defense. Another significant difference is that the attack set U' also appears as the input for the defense (i.e. $\Gamma(U')$). Therefore, while it is easy to find U' to fail $\Gamma(\overline{U})$ for any fixed $\overline{U}$, it is much harder to find a *good direction* to update the attack and converge to *an attack set U^* that fails an entire model space induced by itself:* $\Gamma(U^*)$.

(D3) $\Gamma(U')$ can be a *random variable*. In the classic minimax threat model, the attacker faces a fixed model. However, the output of Γ can be a *random variable of models* due to its private randomness, such as the case of Randomized Smoothing (Cohen et al., 2019). In these cases, successfully attacking a single sample of this random variable does not suffice.

Algorithm 1 Fixed Point Attack (FPA)

Require: A transductive learning algorithm Γ, an optional training dataset D, a clean test set V, an initial model $F^{(0)}$, and an integer parameter $T \geqslant 0$ (the number of iterations).

1: **for** $i = 0, 1, \ldots, T$ **do**

2: Attack the model obtained in the last iteration to get the perturbed set:

$$V^{(i)} = \underset{V' \in N(V)}{\arg\max}\, L_a(F^{(i)}, V') \tag{2.5}$$

where L_a is a loss function. Set $U^{(i)} = V^{(i)}\,|_X$.

3: Run the transductive learning algorithm Γ to get the next model: $F^{(i+1)} = \Gamma(D, U^{(i)})$.

4: **end for**

5: Select the best attack set $U^{(k)}$ as $k = \arg\max_{0 \leqslant i \leqslant T} L(F^{(i+1)}, V^{(i)})$.

6: **return** $U^{(k)}$.

Fixed Point Attack: A first attempt. We adapt previous literature for solving bilevel optimization in deep learning setting (Lorraine and Duvenaud, 2018) (designed for supervised learning). The idea is simple: At iteration $i + 1$, we fix $U^{(i)}$ and model space $F^{(i)} = \Gamma(U^{(i)})$, and construct $U^{(i+1)}$ to fail it. We call this the Fixed Point Attack (FPA) (Algorithm 1), as one hopes that this process converges to a good fixed point U^*. Unfortunately, we found FPA to be weak in experiments. The reason is exactly **(D2)**: $U^{(i+1)}$ failing $F^{(i)}$ may not give any indication that it can also fail $F^{(i+1)}$ induced by itself. Note that transfer attack is a special case of FPA by setting $T = 0$.

2.4.2 Strong adaptive attacks from attacking model spaces

To develop stronger adaptive attacks, we consider a key property of the adversarial attacks: The *transferability of adversarial examples*. Various previous work have identified that adversarial examples transfer (Tramèr et al., 2017; Liu et al., 2016), even across vastly different architectures and models. Therefore, if U' is a good attack set, we would expect that U' also fails $\Gamma(\overline{U})$ for $\overline{U}$ close to U'. This leads to the consideration of the following objective:

$$\max_{U'} \min_{\overline{U} \in \mathcal{N}(U')} L_a(\Gamma(\overline{U}), U'). \tag{2.6}$$

where $\mathcal{N}(\cdot)$ is a neighborhood function (possibly different than N). It induces a family of models $\{\Gamma(\overline{U}) \mid \overline{U} \in \mathcal{N}(U')\}$, which we call a *model space*. (in fact, this can be a family of random variables of models) This can be viewed as a natural *robust* version of (2.3) by considering the transferability of U'. While this is seemingly even harder to solve, it has several benefits: **(1) Considering a model space naturally strengthens FPA.** FPA naturally falls into this formulation as a weak instantiation where we consider a single $\overline{U} = U^{(i)}$. Also, considering a model space gives the attacker more information in dealing with the nondifferentiability of Γ **(D1)**. **(2) It relaxes the attacker-defender constraint (D2).** Perhaps more importantly, for the robust objective, we no longer need the same U' to appear in both defender and attacker. Therefore it gives a natural relaxation which makes attack algorithm design easier.

In summary, while "brittle" U' that does not transfer may indeed exist theoretically, their identification can be challenging algorithmically, and its robust variant provides a natural relaxation considering both algorithmic feasibility and attack strength. This thus leads us to the following principle:

The Principle of Attacking Model Spaces. *An effective adaptive attack against a transductive-learning based defense may need to consider* a model space *induced by a proper neighborhood of* U.

Algorithm 2 GREEDY MODEL SPACE ATTACK (GMSA)

Require: A transductive learning algorithm Γ, an optional training dataset D, a clean test set V, an initial model $F^{(0)}$, and an integer parameter $T \geqslant 0$ (the number of iterations).

1: **for** $i = 0, 1, \ldots, T$ **do**

2: Attack the previous models to get the perturbed set:

$$V^{(i)} = \arg\max_{V' \in N(V)} L_{GMSA}(\{F^{(j)}\}_{j=0}^{i}, V') \tag{2.7}$$

where L_{GMSA} is a loss function. Set $U^{(i)} = V^{(i)}|_X$.

3: Run the transductive learning algorithm Γ to get the next model: $F^{(i+1)} = \Gamma(D, U^{(i)})$.

4: **end for**

5: Select the best attack $U^{(k)}$ as $k = \arg\max_{0 \leqslant i \leqslant T} L(F^{(i+1)}, V^{(i)})$,

6: **return** $U^{(k)}$.

An instantiation: Greedy Model Space Attack (GMSA). We give a simplest possible instantiation of the principle, which we call the *Greedy Model Space Attack* (Algorithm 2). In experiments we use this instantiation to break previous defenses. In this instantiation, the family of model spaces to consider is just all the model spaces constructed in previous iterations. $L_{GMSA}(\{F^{(j)}\}_{j=0}^{i}, V')$ is a loss function that the attacker uses to attack the history model spaces. We consider two instantiations: (1) $L_{GMSA}^{AVG}(\{F^{(j)}\}_{j=0}^{i}, V') = \frac{1}{i+1}\sum_{j=0}^{i} L_a(F^{(j)}, V')$, (2) $L_{GMSA}^{MIN}(\{F^{(j)}\}_{j=0}^{i}, V') = \min_{0 \leqslant j \leqslant i} L_a(F^{(j)}, V')$, where L_{GMSA}^{AVG} gives attack algorithm GMSA-AVG, and L_{GMSA}^{MIN} gives attack algorithm GMSA-MIN. We solve (2.7) via Projected Gradient Decent (PGD). For GMSA-AVG, at the i-th iteration, when applying PGD on the data point $\mathbf{x}$ to generate the perturbation δ, we need to do one backpropagation operation for each model in $\{F^{(j)}\}_{j=0}^{i}$ per PGD step. We do the backpropagation for each model sequentially and then accumulate the gradients to update the perturbation δ since we might not have enough memory to store all the models and compute the gradients at once, especially when i is large. For GMSA-MIN, we find that it requires more PGD steps to solve the attack objective at the i-th iteration where we need to attack $i+1$ models simultaneously. Thus, we scale the number of PGD steps at the i-th iteration by a factor of $i+1$ for GMSA-MIN.

2.5 Empirical Study

This section evaluates various transductive-learning based defenses. Our main findings are: (**1**) The robustness of existing transductive defenses like RMC and DENT is overestimated. Under our evaluation framework, those defenses either have little robustness or have almost the same robustness as that of the static base model. To this end, we note that while AutoAttack is effective in evaluating the robustness of static models, it is not effective in evaluating the robustness of transductive defenses. In contrast, our GMSA attack is a strong baseline for attacking transductive defenses. (**2**) We experimented a novel idea of applying Domain Adversarial Neural Networks (Ajakan et al., 2014), an unsupervised domain adaptation technique (Wilson and Cook, 2020), as a transductive-learning based defense. We show that DANN has nontrivial and even better robustness compared to existing work, under AutoAttack, PGD attack, and FPA attack, even though it is broken by GMSA. (**3**) We report a somewhat surprising phenomenon on *transductive adversarial training*: Adversarially retraining the model *using fresh private randomness on a new batch of test-time data* gives a significant increase in robustness, against all of our considered attacks. (**4**) Finally, we demonstrated that URejectron, while enjoying theoretical guarantees in the bounded-VC dimensions situation, can be broken in natural deep learning settings.

Evaluation framework. For each defense, we report accuracy and robustness. The accuracy is the performance on the clean test inputs, and the robustness is the performance under adversarial attacks. The robustness of transductive defenses is estimated using AutoAttack (AA)[3], PGD attack, FPA, GMSA-MIN and GMSA-AVG. We use PGD attack and AutoAttack in the transfer attack setting for the transductive defense: We generate adversarial examples by attacking a static model (e.g. the base model used by the transductive defense), and then evaluate the transductive defense on the generated adversarial examples. Accuracy and robustness of the static models are also reported for comparison. We always use AutoAttack to estimate the robustness of static models since it is the state-of-the-art for the inductive setting. For

[3]We use the standard version of AutoAttack: `https://github.com/fra31/auto-attack/`.

Dataset	Base Model	Accuracy		Robustness					
		Static	RMC	Static	RMC				
				AA	AA	PGD	FPA	GMSA-AVG	GMSA-MIN
MNIST	Standard	99.50	99.00	0.00	97.70	98.30	0.60	**0.50**	1.10
	Madry et al.	99.60	97.00	87.70	95.70	96.10	59.50	61.40	**58.80**
CIFAR-10	Standard	94.30	93.10	0.00	94.20	97.60	8.50	**8.00**	8.10
	Madry et al.	83.20	90.90	44.30	77.90	71.70	40.80	42.50	**39.60**

Table 2.1: Results of evaluating RMC. We also evaluate the static base model for comparison. **Bold** numbers are worst results.

all experiments, the defender uses his own private randomness, which is different from the one used by the attacker. Without specified otherwise, all reported values are percentages. Below we give details.

Runtime Masking and Cleansing (RMC (Wu et al., 2020b)). RMC adapts the network at test time, and was shown to achieve state-of-the-art robustness under several attacks that are unaware of the defense mechanism (thus these attacks are non-adaptive according to our definition). We follow the setup in Wu et al. (2020b) to perform experiments on MNIST and CIFAR-10 to evaluate the robustness of RMC. On MNIST, we consider L_∞ norm attack with $\epsilon = 0.3$ and on CIFAR-10, we consider L_∞ norm attack with $\epsilon = \frac{8}{255}$. We set $T = 9$ for FPA, GMSA-AVG and GMSA-MIN. The performance of RMC is evaluated on a sequence of test points $\mathbf{x}^{(1)}, \cdots, \mathbf{x}^{(n)}$ randomly sampled from the test dataset. So we have a n-round game. The FPA and GMSA attacks are applied on each round and the initial model $F^{(0)}$ used by the attacks at the $(k+1)$-th round is the adapted model (with calibration in RMC) obtained at the k-th round. To save computational cost, we set $n = 1000$. The robustness of RMC is evaluated on a sequence of adversarial examples $\hat{\mathbf{x}}^{(1)}, \cdots, \hat{\mathbf{x}}^{(n)}$ generated by the attacker on the sequence of test points $\mathbf{x}^{(1)}, \cdots, \mathbf{x}^{(n)}$. We evaluate the robustness of RMC in the non-state leaking setting with private randomness (both are in favor of the defender).

Results. The results are in Table 2.1. RMC with the standard model is already broken by FPA attack (weaker than GSMA). Compared to the defense-unaware AutoAttack, our GMSA-AVG attack reduces the robustness from 97.70% to 0.50% on MNIST and from 94.20% to 8.00% on CIFAR-10. Further, RMC with adversarially

trained model actually provides *worse* adversarial robustness than using adversarial training alone. Under our GMSA-MIN attack, the robustness is reduced from 96.10% to 58.80% on MNIST and from 71.70% to 39.60% on CIFAR-10.

Defensive Entropy Minimization (DENT (Wang et al., 2021a)). DENT performs test-time adaptation, and works for any training-time learner. It was shown that DENT improves the robustness of the state-of-the-art adversarial training defenses by 20+ points absolute against AutoAttack on CIFAR-10 under L_∞ norm attack with $\epsilon = \frac{8}{255}$ (DENT is implemented as a model module, and AutoAttack is directly applied to the module, and we denote this as DENT-AA). Wang et al. also considers adaptive attacks for DENT, such as attacking the static base model using AutoAttack to generate adversarial examples, which is the same as the AutoAttack (AA) in our evaluation.

We evaluate the best version of DENT, called DENT+ in Wang et al., under their original settings on CIFAR-10: DENT is combined with various adversarial training defenses, and only the model adaptation is included without input adaptation. The model is adapted sample-wise for six steps by AdaMod (Ding et al., 2019) with learning rate of 0.006, batch size of 128 and no weight decay. The adaptation objective is entropy minimization with the information maximization regularization. To save computational cost, we only evaluate on 1000 examples randomly sampled from the test dataset. We consider L_∞ norm attack with $\epsilon = \frac{8}{255}$. We set $T = 2$ for FPA, GMSA-AVG and GMSA-MIN. We design loss functions for the attacks to generate adversarial examples with high confidence. For PGD attack, FPA, GMSA-AVG and GMSA-MIN, we use the following loss function to find adversarial examples with high confidence: $L_a(F, V) = \frac{1}{|V|}\sum_{(x,y)\in V} \max_{k\neq y} f(x)_k$, where $f(x)$ is the softmax output of the model F. However, it is hard to optimize this loss function. Thus, we use two alternative loss functions to find adversarial examples. One is the untargeted CW loss (Carlini and Wagner, 2017): $L_a^1(F, V) = \frac{1}{|V|}\sum_{(x,y)\in V} -Z(x)_y + \max_{k\neq y} Z(x)_k$, where $Z(x)$ is the logits of the model F (the output of the layer before the softmax layer). The other is the targeted CW loss: $L_a^2(F, V) = \frac{1}{|V|}\sum_{(x,y)\in V} -Z(x)_y + Z(x)_t$, where t is the targeted label and $t \neq y$. For each attack, we use 14 PGD subroutines to solve its attack objective, including 5 PGD subroutines using the untargeted CW

Base Model	Accuracy		Robustness						
	Static	DENT	Static	DENT					
			AA	DENT-AA	AA	PGD	FPA	GMSA-AVG	GMSA-MIN
Wu et al. (2020a)	85.70	86.10	58.00	78.80	64.40	59.50	**59.30**	59.60	59.60
Carmon et al. (2019)	88.00	87.40	57.30	80.10	61.70	**58.40**	**58.40**	58.50	58.50
Sehwag et al. (2020)	87.30	86.90	54.90	76.50	59.60	**55.80**	**55.80**	**55.80**	**55.80**
Wang et al. (2020)	86.60	85.60	53.60	75.90	61.30	55.90	**55.80**	56.10	56.10
Hendrycks et al. (2019a)	85.80	85.50	51.80	77.20	58.40	**54.20**	54.40	**54.20**	**54.20**
Wong et al. (2020)	81.20	81.00	42.40	69.70	48.90	**44.10**	44.30	44.50	44.30
Ding et al. (2020)	82.40	82.40	39.70	62.80	44.80	39.90	39.40	**39.10**	39.20

Table 2.2: Results of evaluating DENT on CIFAR-10 under the naturally-ordered game. We also evaluate the static base model for comparison. **Bold** numbers are worst results.

Base Model	Robustness					
	Static	DENT				
	AA	AA	PGD	FPA	GMSA-AVG	GMSA-MIN
Wu et al. (2020a)	58.00	58.00	50.40	**50.30**	50.40	50.40
Carmon et al. (2019)	57.30	58.00	**51.80**	**51.80**	**51.80**	**51.80**
Sehwag et al. (2020)	54.90	55.90	50.10	**49.80**	50.00	50.00
Wang et al. (2020)	53.60	55.90	49.00	49.10	48.90	**48.70**
Hendrycks et al. (2019a)	51.80	53.10	**48.10**	48.20	48.30	48.30
Wong et al. (2020)	42.40	44.60	40.10	**39.90**	40.00	40.00
Ding et al. (2020)	39.70	42.10	36.20	35.70	35.30	**35.00**

Table 2.3: Results of evaluating DENT on CIFAR-10 under the adversarially-ordered game. **Bold** numbers are worst results.

loss L_a^1 with different random restarts and 9 PGD subroutines using the targeted CW loss L_a^2 with different targeted labels. So for each clean test input $\mathbf{x}$, these PGD subroutines will return 14 adversarial examples $\mathbf{x}'_1, \ldots, \mathbf{x}'_{14}$. Among these adversarial examples, we select the one that maximizes the attack loss with the loss function $L_a(F, V)$ as the final adversarial example $\mathbf{x}'$ for $\mathbf{x}$. We use the same hyper-parameters for all PGD subroutines: the step size is $\frac{1}{255}$, the number of steps is 100, and the random start is used.

Results. We first evaluate the robustness of DENT under the naturally-ordered game, which is the evaluation setting considered in Wang et al. (2021a). The results in Table 2.2 show that both DENT-AA and AA overestimate the robustness of DENT. Our PGD attack reduces the robustness of DENT to be almost the same as that of

Dataset	Accuracy			Robustness						
	Standard	Madry et al.	DANN	Standard	Madry et al.	DANN				
				AA	AA	AA	PGD	FPA	GMSA-AVG	GMSA-MIN
MNIST	99.42	99.16	99.27	0.00	88.92	97.59	96.66	96.81	79.37	**6.17**
CIFAR-10	93.95	86.06	89.61	0.00	39.49	66.61	60.54	53.98	**5.53**	8.56

Table 2.4: Results of evaluating DANN. **Bold** numbers are worst results.

the static defenses. Further, our FPA, GMSA-AVG and GMSA-MIN have similar performance as the PGD attack. The results show that AutoAttack is not effective in evaluating the robustness of transductive defenses. We then evaluate the robustness of DENT under the adversarially-ordered game where the adversary can choose a "worst-case" order of perturbed data points after receiving a large amount of test data and then sends them in batches one at a time to the defender. Specifically, each time the attacker will generate adversarial examples on up to 256 data points, and then sort the adversarial examples by their labels from lowest to highest, and finally send the sorted adversarial examples in batches one at a time to the defender. The results in Table 2.3 show that under the adversarially-ordered game, we can reduce the robustness of DENT to be lower than that of static base models.

Domain Adversarial Neural Network (DANN (Ajakan et al., 2014)). We consider DANN as a transductive defense for adversarial robustness. We train DANN on the labeled training dataset D (source domain) and unlabeled adversarial test dataset U' (target domain), and then evaluate DANN on U'. For each adversarial set U', we train a new DANN model from scratch. We use the standard model trained on D as the base model for DANN. We perform experiments on MNIST and CIFAR-10 to evaluate the adversarial robustness of DANN. On MNIST, we consider L_∞ norm attack with $\epsilon = 0.3$ and on CIFAR-10, we consider L_∞ norm attack with $\epsilon = 8/255$. We set $T = 9$ for FPA, GMSA-AVG and GMSA-MIN.

Results. The results in Table 2.4 show that DANN has non-trivial robustness under AutoAttack, PGD attack and FPA attack. However, under our GMSA attack, DANN has little robustness.

Transductive Adversarial Training (TADV). We consider a simple but novel transductive-learning based defense called transductive adversarial training: After

Dataset	Accuracy		Robustness					
	Madry et al.	TADV	Madry et al.	TADV				
			AA	AA	PGD	FPA	GMSA-AVG	GMSA-MIN
MNIST	99.01	99.05	86.61	96.07	96.48	95.47	**94.27**	95.48
CIFAR-10	87.69	88.51	45.29	72.12	59.05	58.64	**54.12**	57.77

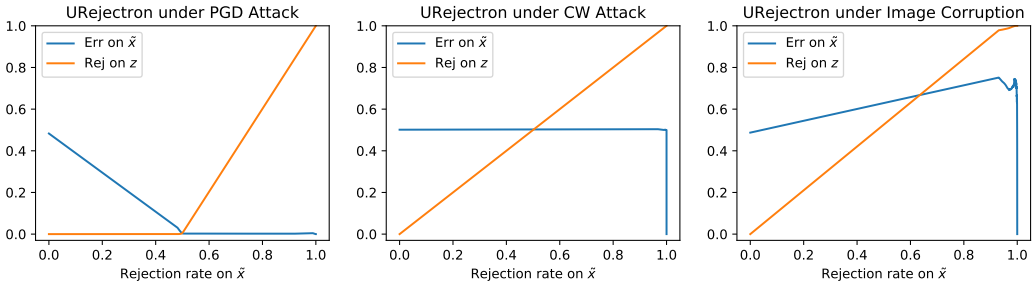
Table 2.5: Results of evaluating TADV. **Bold** numbers are worst results.

receiving a set of examples at the test time, we always adversarially retrain the model using fresh randomness. The key point of this transduction is that *private randomness* is sampled after the attacker's move, and so the attacker cannot directly attack the resulting model as in the inductive case. Specifically, for our GMSA attacks, we attack (with loss L_{GMSA}^{AVG} or L_{GMSA}^{MIN}) an ensemble of $T = 10$ models, adversarially trained with independent randomness, and generate a perturbed test set U'. Then we adversarially train another model from scratch with independent randomness, and check whether U' transfers to the new model (this thus captures the scenario described earlier). We perform experiments on MNIST and CIFAR-10 to evaluate the adversarial robustness of TADV. On MNIST, we consider L_∞ norm attack with $\epsilon = 0.3$ and on CIFAR-10, we consider L_∞ norm attack with $\epsilon = 8/255$. We set $T = 9$ for FPA, GMSA-AVG and GMSA-MIN. Somewhat surprisingly, we find that U' does not transfer very well, and the TADV improves robustness significantly.

Results. The results in Table 2.5 show that transductive adversarial training significantly improves the robustness of adversarial training (Madry et al., 2018). On MNIST, the robustness is improved from 86.61% to 94.27%. On CIFAR-10, the robustness is improved from 45.29% to 54.12%.

URejectron in deep learning settings. URejectron performs transductive learning for defense, and has theoretical guarantees under bounded VC dimension case. We evaluated URejectron on GTSRB dataset using ResNet18 network. We used the same implementation by Goldwasser et al..

Results. Figure 2.1(a) shows that for *transfer attacks* generated by the PGD attack (Madry et al., 2018), URejectron can indeed work as expected. However, by using different attack algorithms, such as CW attack (Carlini and Wagner, 2017), we observe two failure modes: **(1)** *Imperceptible adversarial perturbations that slip*



(a) (b) (c)

Figure 2.1: URejectron in three settings. z contains "normal" examples on which the classifier can have high accuracy. $\tilde{x}$ includes z and consists of a mix of 50% "normal" examples and 50% adversarial examples. In (a), the normal examples are clean test inputs and the adversarial examples are generated by PGD attack. In (b), the "normal" examples are still clean test inputs but adversarial examples are generated by CW attack. In (c), the "normal" examples are generated by image corruptions (adversarial examples are generated by PGD attacks).

through. Figure 2.1(b) shows that one can construct adversarial examples that are very similar to the clean test inputs that can slip through their URejectron construction of S (in the deep learning setting), and cause large errors. **(2)** *Benign perturbations that get rejected*. Figure 2.1(c) shows that we can generate "benign" perturbed examples using image corruptions, such as slightly increased brightness, but URejectron rejects all.

2.6 Conclusion

In this chapter, we formulated threat models for transductive defenses and proposed an attack framework called Greedy Model Space Attack (GMSA) that could serve as a new baseline for evaluating transductive defenses. We showed that GMSA could break previous transductive defenses, which were resilient to previous attacks such as AutoAttack. On the positive side, we showed that transductive adversarial training gave a significant increase in robustness against attacks we considered. For future work, one could explore transductive defenses that could be robust under our GMSA attacks, and could also explore even stronger adaptive attacks that were

effective in evaluating transductive defenses.

3 STRATIFIED ADVERSARIAL ROBUSTNESS WITH REJECTION

Contribution statement. This chapter is joint work with Jayaram Raghuram, Jihye Choi, Xi Wu, Yingyu Liang, and Somesh Jha. The author Jiefeng Chen proposed the method, contributed to part of the theoretical analysis, and completed all the experiments. The paper version of this chapter appeared in ICML 2023 (Chen et al., 2023a).

3.1 Introduction

Building robust models against adversarial attacks is critical for designing secure and reliable machine learning systems (Biggio et al., 2013b; Szegedy et al., 2014; Biggio and Roli, 2018; Madry et al., 2018; Zhang et al., 2019). However, the robust error of existing methods on complex datasets is still not satisfactory (e.g., Croce et al. (2020)). Also, the robust models usually have poor generalization to threat models that are not utilized during training (Stutz et al., 2020; Laidlaw et al., 2021). Given these limitations, it is important to design selective classifiers that know when to reject or abstain from predicting on adversarial examples. This can be especially crucial when it comes to real-world, safety-critical systems such as self-driving cars, where abstaining from prediction is often a much safer alternative than making an incorrect decision. Along this line of adversarial robustness with rejection, several recent studies (Laidlaw and Feizi, 2019b; Stutz et al., 2020; Sheikholeslami et al., 2021; Pang et al., 2022; Tramèr, 2021; Kato et al., 2020) have extended the standard definition of robust error to the setting where the classifier can also reject inputs, and they consider rejecting any perturbed input to be a valid decision that does not count towards the robust error.

A key limitation with these studies is that they associate *zero cost* with the rejection decision on perturbed inputs [1], whereas rejection can often have a high cost in many practical applications. For example, consider a selective classifier for

[1] Note that the rejection of clean inputs does incur a cost.

traffic signs in a self-driving system. If it rejects an input (e.g., a perturbed "speed limit 60" sign), then the system may not know how to react and thus need human intervention (e.g., adjust the speed). In such cases, rejection has the cost of service-denial and manual intervention (Markoff, 2016; Cunningham and Regan, 2015; Mozannar and Sontag, 2020). In contrast to the practical consideration, existing studies on adversarial robustness with rejection typically do not explicitly consider a cost for rejecting perturbed inputs. The learned models thus may not satisfy the need of these applications. Indeed, the models from existing methods may end up rejecting too many slightly-perturbed inputs that could be correctly classified. As a concrete example, on MNIST with ℓ_∞-perturbation magnitude 0.4, the method CCAT (Stutz et al., 2020) achieves very good performance on the existing metrics such as 1.82% rejection rate on clean test inputs and 75.50% robust accuracy with detection (a metric introduced in (Tramèr, 2021); see Eq. 3.2). However, for 99.30% of the test points, CCAT will reject some small perturbations within magnitude as small as 0.02. More results can be found in our experiments in Section 3.6. In summary, while rejecting such small perturbations has a cost, existing studies have not adequately measured the quality of the selective classifiers, and the training methods may not learn desired models for such applications.

To address this limitation, we revisit adversarially-robust classification with rejection by introducing rejection loss functions to model the potential cost of rejection. This offers a more flexible framework than the traditional adversarial robustness studies that do not consider rejection (roughly equivalent to associating with rejection the same loss as mis-classification), and recent studies on adversarial robustness with rejection (e.g. (Tramèr, 2021)) that associate zero loss with rejecting any perturbed input. We focus on *the stratified rejection setting where the rejection loss functions are monotonically non-increasing in the perturbation magnitude*. This is motivated by the consideration that small input perturbations should not be rejected when correct classification is possible, and thus their rejection should incur a large cost. However, large input perturbations can often be harder to classify, and rejection may be the best option when correct classification is not possible; so they should incur a lower rejection cost compared to smaller perturbations. Furthermore,

we consider the challenging scenario where *the rejection loss function used for testing is unknown at training time*. That is, the learning method is not designed for one specific rejection loss; the learned selective classifier should work under a range of reasonable rejection loss functions. Our goal is then to design such a method that can learn a robust selective classifier with small loss due to *both* mis-classification and rejection.

In summary, we make the following contributions:

- We propose to introduce rejection loss functions to model the potential cost of rejection in applications, and study the stratified rejection setting with monotonically non-increasing rejection loss functions (Section 3.3).
- We provide a theoretical analysis of the stratified rejection setting. We analyze the existence of a robust selective classifier and discuss conditions when it can improve over classifiers without rejection (Section 3.4).
- We propose a novel defense method CPR inspired by our theoretical analysis. Our experiments demonstrate that CPR significantly outperforms previous methods under strong adaptive attacks (Sections 3.5 and 3.6). Its performance is strong for different rejection losses, on both traditional and our new metrics, and under seen and unseen attacks. CPR can be combined with different kinds of adversarial training methods (e.g., TRADES (Zhang et al., 2019)) to enhance their robustness.

3.2 Related Work

Adversarial robustness of deep learning models has received significant attention in recent years. Many defenses have been proposed and most of them have been broken by strong adaptive attacks (Athalye et al., 2018; Tramèr et al., 2020). The most effective approach for improving adversarial robustness is adversarial training (Madry et al., 2018; Zhang et al., 2019). However, adversarial training still does not achieve very high robust accuracy on complex datasets. For example, as

reported in RobustBench (Croce et al., 2020), even state-of-the-art adversarially trained models struggle to exceed 67% robust test accuracy on CIFAR-10.

One approach to break this adversarial robustness bottleneck is to allow the classifier to reject inputs, instead of trying to correctly classify all of them. Standard (non-adversarial) classification with a reject option (or selective classification) has been extensively studied in the literature (Tax and Duin, 2008; Geifman and El-Yaniv, 2019; Charoenphakdee et al., 2021; Cortes et al., 2016). Selective classification in the transductive setting with provable guarantees has been studied by Goldwasser et al. (2020). Recently, there has been a great interest in exploring adversarially robust classification with a reject option (Laidlaw and Feizi, 2019b; Stutz et al., 2020; Kato et al., 2020; Yin et al., 2020; Sheikholeslami et al., 2021; Tramèr, 2021; Pang et al., 2022; Balcan et al., 2023). We next discuss some of these closely related works.

Stutz et al. (2020) proposed to adversarially train confidence-calibrated models using label smoothing and confidence thresholding so that they can generalize to unseen adversarial attacks. Sheikholeslami et al. (2021) modified existing certified-defense mechanisms to allow the classifier to either robustly classify or detect adversarial attacks, and showed that it can lead to better certified robustness, especially for large perturbation sizes. Pang et al. (2022) observed that two coupled metrics, the prediction confidence and the true confidence (T-Con), can be combined to provably distinguish correctly-classified inputs from mis-classified inputs. Based on this, they propose to learn a rectified confidence (R-Con) that models T-Con, which is then used to adversarially train a selective classifier. Laidlaw and Feizi (2019b) proposed a method called Combined Abstention Robustness Learning (CARL) for jointly learning a classifier and the region of the input space on which it should abstain, and showed that training with CARL can result in a more accurate and robust classifier. In Balcan et al. (2023), the authors introduced a random feature subspace threat model and showed that classifiers without the ability to abstain (reject) are provably vulnerable to this adversary; but allowing the classifier to abstain (e.g., via a thresholded nearest-neighbor algorithm) can overcome such attacks.

An important aspect that has not been explored in prior works is the cost or loss

of rejecting perturbed inputs. This is important for designing robust classifiers that do not reject many slightly-perturbed inputs which could be correctly classified. To the best of our knowledge, we are the first to study adversarially-robust classification with rejection in the stratified-rejection setting.

3.3 Stratified Adversarial Robustness with Rejection

Notations. Let $\mathcal{X} \subseteq \mathbb{R}^d$ denote the space of inputs $\mathbf{x}$ and $\overline{\mathcal{Y}} := \{1, \cdots, k\}$ denote the space of outputs y. Let $\mathcal{Y} := \overline{\mathcal{Y}} \cup \{\perp\}$ be the extended output space where $\perp$ denotes the abstain or rejection option. Let Δ_k denote the set of output probabilities over $\overline{\mathcal{Y}}$ (i.e., the simplex in k-dimensions). Let $d(\mathbf{x}, \mathbf{x}')$ be a norm-induced distance on $\mathcal{X}$ (e.g., the ℓ_p-distance for some $p \geqslant 1$), and let $\mathcal{N}(\mathbf{x}, r) := \{\mathbf{x}' \in \mathcal{X} : d(\mathbf{x}', \mathbf{x}) \leqslant r\}$ denote the neighborhood of $\mathbf{x}$ with distance r. Let $\wedge$ and $\vee$ denote the min and max operations respectively (reducing to AND and OR operations when applied on boolean values). Let $\mathbf{1}\{c\}$ define the binary indicator function which takes value 1 (0) when the condition c is true (false).

Review. We first review the standard setting of adversarial robustness without rejection and the setting with rejection in a recent line of work. Given samples from a distribution $\mathcal{D}$ over $\mathcal{X} \times \overline{\mathcal{Y}}$, the goal is to learn a classifier with a reject option (a selective classifier), $f : \mathcal{X} \to \mathcal{Y}$, that has a small error. The *standard robust error* at adversarial budget $\epsilon > 0$ is defined as Madry et al. (2018):

$$R_\epsilon(f) := \mathop{\mathbb{E}}_{(\mathbf{x},y)\sim\mathcal{D}} \left[\max_{\mathbf{x}' \in \mathcal{N}(\mathbf{x},\epsilon)} \mathbf{1}\{f(\mathbf{x}') \neq y\} \right], \tag{3.1}$$

which does not allow rejection (i.e., rejection is an error). A few recent studies (e.g. Tramèr (2021)) have proposed the *robust error with detection* at adversarial budget ϵ as

$$R^{\text{rej}}_\epsilon(f) := \mathop{\mathbb{E}}_{(\mathbf{x},y)\sim\mathcal{D}} \left[\mathbf{1}\{f(\mathbf{x}) \neq y\} \vee \max_{\mathbf{x}' \in \mathcal{N}(\mathbf{x},\epsilon)} \mathbf{1}\big\{f(\mathbf{x}') \notin \{y, \perp\}\big\} \right], \tag{3.2}$$

which allows the rejection of small (even infinitesimal) perturbations without incurring any error.

Rejection Loss Functions. The above studies are not well-suited for the needs of certain applications where rejection can have a cost. We would like to associate with the reject decision a loss that is a function of the perturbation magnitude. Intuitively, rejecting a clean input should incur the maximum loss, and the loss of rejecting a perturbed input should decrease (or at least not increase) as the perturbation magnitude increases. Formally, let $\ell^{\text{rej}}(r) : [0, \infty) \to [0, 1]$ be a function specifying the loss of rejecting a perturbation $\mathbf{x}'$ of a clean input $\mathbf{x}$ with perturbation magnitude $r = d(\mathbf{x}, \mathbf{x}')$. We consider two concrete cases of such losses. The *step rejection loss* is defined as

$$\ell^{\text{rej}}(r) \;=\; \mathbf{1}\{r \leqslant \alpha_0 \epsilon\} \tag{3.3}$$

for some $\alpha_0 \in [0, 1]$. That is, rejecting a perturbed input of magnitude smaller than $\alpha_0 \epsilon$ has a loss 1 but rejecting larger perturbations has no loss. The *ramp rejection loss* is defined as follows for some $t \geqslant 0$

$$\ell^{\text{rej}}(r) \;=\; \left(1 - \frac{r}{\epsilon}\right)^t. \tag{3.4}$$

For instance, for $t = 1$, the ramp rejection loss decreases linearly with the perturbation magnitude.

Total Robust Loss. With the rejection loss modeling the potential cost of rejection, the adversary can make the selective classifier suffer a mis-classification loss when there exists a perturbation $\mathbf{x}'$ with $f(\mathbf{x}') \notin \{y, \perp\}$, or suffer a rejection loss $\ell^{\text{rej}}(d(\mathbf{x}, \mathbf{x}'))$ when there exists a perturbation $\mathbf{x}'$ that is rejected (i.e., $f(\mathbf{x}') = \perp$). Then our goal is to learn a selective classifier that has a small total loss due to both mis-classification and rejection induced by the adversary, which is formalized as follows.

Definition 3.1. The total robust loss *of a selective classifier* f *at adversarial budget* $\epsilon > 0$ *with respect to a given rejection loss function* ℓ^{rej} *is:*

$$L_\epsilon(f; \ell^{rej}) := \mathop{\mathbb{E}}_{(x,y)\sim\mathcal{D}} \left[\max_{x' \in \mathcal{N}(x,\epsilon)} \Big(\mathbf{1}\big\{ f(x') \notin \{y, \perp\} \big\} \vee \mathbf{1}\{f(x') = \perp\}\, \ell^{rej}(d(x, x')) \Big) \right].$$

Here $\vee$ denotes the maximum of the mis-classification loss and the rejection loss. While this definition is compatible with any rejection loss, we will focus on the monotonically non-increasing ones. This definition also applies to classifiers without a rejection option, in which case the total robust loss reduces to the standard robust error $R_\epsilon(f)$.

The Curve of Robust Error. The definition of the total robust loss however depends on the specific instantiation of ℓ^{rej}, which may vary for different applications. In practice, we would like to have a single evaluation of f which can be combined with different definitions of ℓ^{rej} to compute the total robust loss. Towards this end, we propose the notion of the *curve of robust error (or accuracy)*, which generalizes the existing metrics. More importantly, Lemma 3.3 shows that the curve can be used to compute the total robust loss for different rejection losses (proof given in Appendix A.1.2).

Definition 3.2. The curve of robust error *of a selective classifier* f *at adversarial budget* $\epsilon \geqslant 0$ *is* $\{R^{rej}_\epsilon(f, \alpha) : \alpha \in [0, 1]\}$, *where*

$$R^{rej}_\epsilon(f, \alpha) := \mathop{\mathbb{E}}_{(x,y)\sim\mathcal{D}} \left[\max_{x' \in \mathcal{N}(x,\, \alpha\epsilon)} \mathbf{1}\{f(x') \neq y\} \vee \max_{x'' \in \mathcal{N}(x,\epsilon)} \mathbf{1}\big\{ f(x'') \notin \{y, \perp\} \big\} \right]. \quad (3.5)$$

The curve of robust accuracy *or simply* the robustness curve *of* f *at adversarial budget* ϵ *is defined as*

$$\{A^{rej}_\epsilon(f, \alpha) := 1 - R^{rej}_\epsilon(f, \alpha) : \alpha \in [0, 1]\}. \quad (3.6)$$

We name $R_\epsilon^{\text{rej}}(f, \alpha)$ the *robust error with rejection* at α, and $A_\epsilon^{\text{rej}}(f, \alpha)$ the *robustness with rejection* at α. The intuition behind $R_\epsilon^{\text{rej}}(f, \alpha)$ for a fixed α is clear. For small perturbations within $\mathcal{N}(\mathbf{x}, \alpha\epsilon)$, both an incorrect prediction and rejection are considered an error. For larger perturbations outside $\mathcal{N}(\mathbf{x}, \alpha\epsilon)$, rejection is *not* considered to be an error (i.e., the classifier can either classify correctly or reject larger perturbations). Moreover, $R_\epsilon^{\text{rej}}(f, \alpha)$ actually includes several existing metrics as special cases:

- When $\alpha = 1$, the metric reduces to the standard robust error at budget ϵ in Eq. (3.1), i.e., $R_\epsilon^{\text{rej}}(f, 1) = R_\epsilon(f)$.
- When $\alpha = 0$, it reduces to the robust error with detection at budget ϵ in Eq. (3.2), i.e., $R_\epsilon^{\text{rej}}(f, 0) = R_\epsilon^{\text{rej}}(f)$. Here rejection incurs an error *only* for clean inputs.
- For any classifier f *without rejection* and any $\alpha \in [0, 1]$, it reduces to the standard robust error at budget ϵ defined in Eq. (3.1), i.e., $R_\epsilon^{\text{rej}}(f, \alpha) = R_\epsilon(f)$.

Lemma 3.3. *Let* $s(\alpha) := R_\epsilon^{rej}(f, \alpha)$. *Suppose the rejection loss* $\ell^{rej} : [0, \infty) \mapsto [0, 1]$ *is monotonically non-increasing, differentiable almost everywhere, and* $\ell^{rej}(0) = 1$ *and* $\ell^{rej}(\epsilon) = 0$. *Then the total robust loss simplifies to*

$$L_\epsilon(f; \ell^{rej}) \;=\; -\int_0^1 s(\alpha)\, \mathrm{d}\ell^{rej}(\alpha\epsilon). \tag{3.7}$$

Given this nice connection between the total robust loss and the curve of robust error, we advocate for evaluating a selective classifier f using the curve $\{R_\epsilon^{\text{rej}}(f, \alpha) : \alpha \in [0, 1]\}$. Without knowing the concrete definition of the rejection loss ℓ^{rej}, this curve can give a holographic evaluation of the model. Even when f is trained with a specific rejection loss ℓ^{rej} in mind, the curve is helpful in providing a more complete evaluation w.r.t. *any other* definition of the rejection loss at deployment time.

3.4 Theoretical Analysis

Our goal is to learn a robust selective classifier, even when the training does not know the precise specification of the rejection loss for testing. Some fundamental questions arise:

Q1. *Whether and under what conditions does there exist a selective classifier with a small total robust loss?* Many applications could have a cost for certain rejections, e.g., rejecting very small perturbations is undesirable. To design algorithms for such applications, we would like to first investigate the existence of a solution with a small total robust loss.

Q2. *Can allowing rejection lead to a smaller total robust loss?* The question is essentially about the benefit of selective classification over traditional adversarial robustness (without rejection) that tries to correctly classify all perturbations, typically by adversarial training or its variants.

The following theorem helps address these questions: by allowing rejection, there exists a selective classifier with a small total robust loss under proper conditions.

Theorem 3.4. *Consider binary classification. Let* $f^*(\mathbf{x})$ *be any classifier without a rejection option. For any* $\delta \in [0,1]$ *and* $\epsilon \geqslant 0$*, there is a selective classifier* f_δ *whose robust error curve is bounded by:*

$$R_\epsilon^{rej}(f_\delta, \alpha) \leqslant R_{\epsilon'}(f^*), \ \forall \alpha \in [0,1] \tag{3.8}$$

where $\epsilon' = \max\{(\alpha+\delta)\epsilon, (1-\delta)\epsilon\}$*. Moreover, the bound is tight: for any* $\alpha \in [0,1]$*, there exist simple data distributions and* f^* *such that any* f *must have* $R_\epsilon^{rej}(f, \alpha) \geqslant R_{\epsilon'}(f^*)$*.*

Proof Sketch: For any $r \geqslant 0$, let $\mathcal{N}(f^*, r)$ denote the region within distance r to the decision boundary of f^*: $\mathcal{N}(f^*, r) := \{\mathbf{x} \in \mathcal{X} : \exists \mathbf{x}' \in \mathcal{N}(\mathbf{x}, r), f^*(\mathbf{x}') \neq f^*(\mathbf{x})\}$. Consider

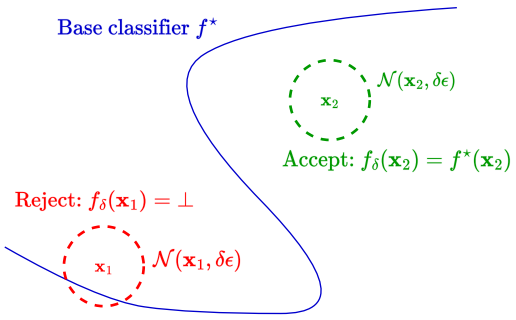


Figure 3.1: Proposed construction of a selective classifier f_δ from the base classifier f^* for Theorem 3.4. Input $\mathbf{x}_1$ is close to the boundary of f^* and is rejected, while input $\mathbf{x}_2$ is accepted.

a parameter $\delta \in [0, 1]$ and construct f_δ as follows:

$$f_\delta(\mathbf{x}) := \begin{cases} \perp & \text{if } \mathbf{x} \in \mathcal{N}(f^*, \delta\epsilon), \\ f^*(\mathbf{x}) & \text{otherwise.} \end{cases} \tag{3.9}$$

This is illustrated in Fig. 3.1. We will show that any clean data $(\mathbf{x}, y)$ contributing error in $R_\epsilon^{\text{rej}}(f_\delta, \alpha)$ must contribute error in $R_{\epsilon'}(f^*)$, so $R_\epsilon^{\text{rej}}(f_\delta, \alpha) \leqslant R_{\epsilon'}(f^*)$. Intuitively, f_δ and f^* differ on the region $\mathcal{N}(f^*, \delta\epsilon)$. If f_δ gets a loss on a (possibly perturbed) input $\mathbf{x}' \in \mathcal{N}(f^*, \delta\epsilon)$, then the original input $\mathbf{x}$ is close to $\mathbf{x}'$ and thus close to the boundary of f^*. Therefore, $\mathbf{x}$ can be perturbed to cross the boundary and contribute error in $R_{\epsilon'}(f^*)$. The complete proof is given in Appendix A.1.3. □

Condition for Successful Selective Classification. The theorem shows that when the data allows a small robust error at budget $\epsilon' = \max\{(\alpha + \delta)\epsilon, (1 - \delta)\epsilon\}$, there is a selective classifier f_δ with a small robust error with rejection $R_\epsilon^{\text{rej}}(f_\delta, \alpha)$, which is bounded by the small robust error $R_{\epsilon'}(f^*)$. This shows a trade-off between the performance for small and large α. For $\alpha < 1 - \delta$, we have $\epsilon' < \epsilon$ and $R_\epsilon^{\text{rej}}(f_\delta, \alpha) \leqslant R_{\epsilon'}(f^*) \leqslant R_\epsilon(f^*)$. Note that $R_\epsilon(f^*) = R_\epsilon^{\text{rej}}(f^*, \alpha)$, so $R_\epsilon^{\text{rej}}(f_\delta, \alpha) \leqslant R_\epsilon^{\text{rej}}(f^*, \alpha)$, i.e., f_δ can be better than f^* for such an α. However, for $\alpha \geqslant 1 - \delta$, the theorem does not guarantee $R_\epsilon^{\text{rej}}(f_\delta, \alpha) \leqslant R_\epsilon(f^*)$, i.e., f_δ may be worse than f^* for such an α. This

trade-off is necessary in the worst case since the bound is tight, and is also observed in our experiments in Section 3.6. It is favorable towards a lower total robust loss when the rejection loss is monotonically non-increasing, as discussed in detail below.

Benefit of Selective Classification. The trade-off discussed above shows that by allowing rejection, the selective classifier f_δ can potentially have a smaller total robust loss than the classifier f^* without rejection. When $\alpha < 1 - \delta$, we have $R_\epsilon^{\text{rej}}(f_\delta, \alpha) \leqslant R_{\epsilon'}(f^*) \leqslant R_\epsilon(f^*) = R_\epsilon^{\text{rej}}(f^*, \alpha)$. There can be a big improvement: when a large fraction of correctly-classified clean inputs have distances in (ϵ', ϵ) to the boundary of f^*, we have $R_{\epsilon'}(f^*) \ll R_\epsilon(f^*)$ and thus $R_\epsilon^{\text{rej}}(f_\delta, \alpha) \ll R_\epsilon^{\text{rej}}(f^*, \alpha)$. This can lead to a significant improvement in the total robust loss. When $\alpha \geqslant 1 - \delta$, $R_\epsilon^{\text{rej}}(f, \alpha)$ may be worse than $R_\epsilon^{\text{rej}}(f^*, \alpha)$, but that only leads to milder rejection loss when ℓ^{rej} is monotonically decreasing. Then overall, the total robust loss can be improved, while the decreased amount would depend on the concrete data distribution and rejection loss.

Theorem A.3 in Appendix A.1.4 provides a rigorous and fine-grained analysis of the robustness curve and the total robust loss of f_δ. In general, the total robust losses of f_δ and f^* only differ on points with distances larger than $(1 - \delta)\epsilon$ from the boundary of f^*. For such points, f_δ can correctly classify all their small perturbations of magnitude at most $\epsilon_0 := (1 - 2\delta)\epsilon$, and reject or correctly classify their larger perturbations. Intuitively, it correctly classifies small perturbations and rejects or correctly classifies large perturbations. Then f_δ only gets rejection loss for large magnitudes, and thus potentially gets smaller total robust losses than f^* for monotonically non-increasing rejection losses.

For a concrete example, suppose f^* has 0 standard error on the data distribution and each data point has distance at least $\frac{3\epsilon}{4}$ to the decision boundary of f^*. Then when $\delta = \frac{1}{4}$, for any data point, f_δ can correctly classify all the small perturbations with magnitude bounded by $\frac{\epsilon}{2}$, and can reject or correctly classify the larger perturbations. Then $R_\epsilon^{\text{rej}}(f_\delta, \alpha) = 0$ for $\alpha \leqslant \frac{1}{2}$. For any step rejection loss with parameter $\alpha_0 \leqslant \frac{1}{2}$, the total robust loss of f_δ is 0, i.e., this fixed f_δ can work well for a wide

range of step rejection losses. In contrast, the total robust loss of f^* is as large as $R_\epsilon(f^*)$, which is the probability mass of points within ϵ distance to the boundary of f^*. If there is a large mass of points with distance in $[\frac{3\epsilon}{4}, \epsilon]$, then f_δ has a significant advantage over f^*.

3.5 Proposed Defense

We propose a defense method called adversarial training with consistent prediction-based rejection (CPR), following our theoretical analysis. CPR aims to learn the f_δ in our analysis, where PGD is used to check the condition for rejection efficiently at test time. Our defense is quite general in that it can take any base classifier f^* to construct the selective classifier f_δ. Finally, to evaluate the robustness of the defense, we also discuss the design of adaptive attacks.

3.5.1 Consistent Prediction-Based Rejection

The CPR defense is essentially the selective classifier f_δ in Eq. (3.9) in Theorem 3.4: given a base classifier and an input $\mathbf{x}$, we define a selective classifier that rejects the input whenever the predictions in a small neighborhood of $\mathbf{x}$ are not consistent; otherwise it returns the class prediction of $\mathbf{x}$. Equivalently, it rejects inputs within a small distance to the decision boundary of the given base classifier.

To formalize the details, we introduce some notations. Consider a base classifier without rejection (i.e., f^*) realized by a network with parameters θ, whose predicted class probabilities are $\mathbf{h}(\mathbf{x}\,;\theta) = [h_1(\mathbf{x}\,;\theta), \cdots, h_k(\mathbf{x}\,;\theta)] \in \Delta_k$. The class prediction is $\widehat{y}(\mathbf{x}) := \operatorname{argmax}_{y \in \overline{\mathcal{Y}}} h_y(\mathbf{x}\,;\theta)$, and the cross-entropy loss is $\ell_{CE}(\mathbf{x}, y) = -\log h_y(\mathbf{x}\,;\theta)$. CPR aims to learn a selective classifier (corresponding to f_δ):

$$\mathbf{g}_{\widetilde{\epsilon}}(\mathbf{x}\,;\theta) = \begin{cases} \mathbf{h}(\mathbf{x}\,;\theta) & \text{if } \max\limits_{\widetilde{\mathbf{x}} \in \mathcal{N}(\mathbf{x}, \widetilde{\epsilon})} \mathbf{1}\{\widehat{y}(\widetilde{\mathbf{x}}) \neq \widehat{y}(\mathbf{x})\} = 0 \\ \perp & \text{otherwise,} \end{cases} \tag{3.10}$$

Algorithm 3 CONSISTENT PREDICTION-BASED REJECTION

Require: A base model $\mathbf{h}$, a test input $\mathbf{x}$ (potentially with adversarial perturbations), the consistency radius $\widetilde{\epsilon} > 0$, the number of PGD steps $m \geqslant 1$, and the PGD step size $\eta > 0$.

1: Class prediction: $y = \widehat{y}(\mathbf{x}) = \arg\max_{y' \in \overline{y}} h_{y'}(\mathbf{x}\,;\theta)$
2: Initialize the adversarial input: $\mathbf{x}_0 = \mathbf{x}$
3: **for** $i = 1, 2, \ldots, m$ **do**
4: $\quad \widehat{\mathbf{x}}_i = \mathbf{x}_{i-1} + \eta\, \mathrm{sign}\big(\nabla_{\mathbf{x}_{i-1}} \ell_{CE}(\mathbf{x}_{i-1}, y)\big)$
5: $\quad \mathbf{x}_i = \mathrm{Proj}\big(\widehat{\mathbf{x}}_i, \mathcal{N}(\mathbf{x}, \widetilde{\epsilon})\big)$ {Project $\widehat{\mathbf{x}}_i$ to $\mathcal{N}(\mathbf{x}, \widetilde{\epsilon})$}
6: **end for**
7: Define $T(\mathbf{x}) := \mathbf{x}_m$

Ensure: If $\arg\max_{y' \in \overline{y}} h_{y'}(T(\mathbf{x})\,;\theta) \neq y$, reject $\mathbf{x}$; otherwise, accept $\mathbf{x}$ and output the predicted class y.

where $\widetilde{\epsilon} > 0$ is a hyper-parameter we call the *consistency radius*. To check the consistent-prediction condition efficiently, we use the projected gradient descent (PGD) method (Madry et al., 2018) to find the worst-case perturbed input $\widetilde{\mathbf{x}}$. The selective classifier is then redefined as:

$$\mathbf{g}_{\widetilde{\epsilon}}(\mathbf{x}\,;\theta) = \begin{cases} \mathbf{h}(\mathbf{x}\,;\theta) & \text{if } \widehat{y}(\widetilde{\mathbf{x}}) = \widehat{y}(\mathbf{x}) \\ \perp & \text{otherwise,} \end{cases} \tag{3.11}$$

$$\text{where } \widetilde{\mathbf{x}} = \underset{\widetilde{\mathbf{x}} \in \mathcal{N}(\mathbf{x}, \widetilde{\epsilon})}{\arg\max}\ \ell_{CE}(\widetilde{\mathbf{x}}, \widehat{y}(\mathbf{x})). \tag{3.12}$$

Then $\widehat{y}(\mathbf{x})$ is the predicted class when the input is accepted. The details of our CPR defense are given in Algorithm 3. Note that the PGD method in the algorithm is deterministic. Thus, the mapping $T(\mathbf{x}) = \mathbf{x}_m$ is deterministic and we can summarize the defense as: if $\widehat{y}(T(\mathbf{x})) \neq \widehat{y}(\mathbf{x})$, then reject $\mathbf{x}$; otherwise accept $\mathbf{x}$ and output the predicted class $\widehat{y}(\mathbf{x})$.

The robust base model for our defense $\mathbf{h}$ can be trained using methods such as standard adversarial training (AT) (Madry et al., 2018), TRADES (Zhang et al., 2019), and MART (Wang et al., 2020) (we will focus on the first two methods). An advantage of the proposed defense is that it is agnostic of the rejection loss function used for evaluation, and therefore can generalize to multiple choices of the rejection

loss function (see Section 3.6). We note that prior works such as Laidlaw and Feizi (2019b) and Tramèr (2021) have also proposed the idea of using the consistency check for rejecting inputs to a classifier (equivalent to rejecting inputs based on their distance to the decision boundary). However, we are the first to systematically implement and evaluate such a defense, and propose strong adaptive attacks for evaluating it.

3.5.2 Adaptive Attacks

In this section, we design principled adaptive attacks to evaluate the proposed defense. By Definition 3.2 and Proposition 8, in order to compute the robustness curve, we need to design both inner and outer attacks and then combine them to get the final attack. We design multiple inner and outer attacks sketched below and ensemble them to get the strongest final attack. More details can be found in Appendix A.3

Inner Attack. For a given $\alpha \in [0, 1]$ on the robustness curve, the goal of the inner attack is to find an input $\mathbf{x}' \in \mathcal{N}(\mathbf{x}, \alpha\epsilon)$ that is rejected. For CPR, this translates to finding $\mathbf{x}' \in \mathcal{N}(\mathbf{x}, \alpha\epsilon)$ such that the base model has different predictions on $\mathbf{x}'$ and $T(\mathbf{x}')$.

Our first attack is Low-Confidence Inner Attack (**LCIA**), which finds $\mathbf{x}'$ by minimizing the confidence of the base model within $\mathcal{N}(\mathbf{x}, \alpha\epsilon)$. Recall that the mapping $T(\mathbf{x}')$ attempts to minimize the base model's probability on the predicted class $\widehat{y}(\mathbf{x}')$. So, if the base model has low confidence on $\mathbf{x}'$, then it will very likely have even lower probability for $\widehat{y}(\mathbf{x}')$ on $T(\mathbf{x}')$, and thus have different predictions on $T(\mathbf{x}')$ and $\mathbf{x}'$. The attack objective in this case is $\mathbf{x}' = \arg\max_{\mathbf{z} \in \mathcal{N}(\mathbf{x}, \alpha\epsilon)} -\log h_{\max}(\mathbf{z}; \theta)$, where $h_{\max}(\mathbf{z}; \theta) = \max_{y \in \overline{y}} h_y(\mathbf{z}; \theta)$ is the prediction confidence. We use the temperature-scaled log-sum-exponential approximation to the max function in order to make it differentiable. We also consider a variant of LCIA, named Consistent-Low-Confidence Inner Attack (**CLCIA**), that minimizes the confidence of the base model on both $\mathbf{x}'$ and $T(\mathbf{x}')$ (details in Appendix A.3).

The third attack is Prediction-Disagreement Inner Attack (**PDIA**), an adaptive multi-target attack based on the BPDA method (Athalye et al., 2018). We use BPDA since $T(\mathbf{x})$ does not have a closed-form expression and is not differentiable. This attack considers all possible target classes and attempts to find $\mathbf{x}'$ such that the base model has high probability for the target class at $\mathbf{x}'$ and a low probability for the target class at $T(\mathbf{x}')$ (thereby encouraging rejection). The attack objective is: for each target class $j = 1, \ldots, k$,

$$\mathbf{x}'_j = \underset{\mathbf{z} \in \mathcal{N}(\mathbf{x}, \alpha\epsilon)}{\arg\max} \Big[\log h_j(\mathbf{z}\,; \theta) \,-\, \log h_j(T(\mathbf{z})\,; \theta) \Big].$$

Then we select the strongest adversarial example $\mathbf{x}'$ from the multiple target classes as follows:

$$\mathbf{x}' \,=\, \mathbf{x}'_{j^\star} \quad \text{where} \quad j^\star \,=\, \underset{j \in [k]}{\arg\max} \Big[\log h_j(\mathbf{x}'_j\,; \theta) \,-\, \log h_j(T(\mathbf{x}'_j)\,; \theta) \Big]. \tag{3.13}$$

Outer Attack. Given a clean input $(\mathbf{x}, y)$, the goal of the outer attack is to find $\mathbf{x}'' \in \mathcal{N}(\mathbf{x}, \epsilon)$ such that the base model has a consistent incorrect prediction with high confidence on both $\mathbf{x}''$ and $T(\mathbf{x}'')$ (ensuring that $\mathbf{x}''$ is accepted and misclassified). We propose the Consistent High-Confidence Misclassification Outer Attack (**CHCMOA**), an adaptive multi-target attack based on BPDA. The attack objective is: for each target class $j \in [k] \setminus \{y\}$,

$$\mathbf{x}''_j = \underset{\mathbf{z} \in \mathcal{N}(\mathbf{x}, \epsilon)}{\arg\max} \Big[\log h_j(\mathbf{z}\,; \theta) \,+\, \log h_j(T(\mathbf{z})\,; \theta) \Big].$$

Then we select the strongest adversarial example $\mathbf{x}''$ via:

$$\mathbf{x}'' \,=\, \mathbf{x}''_{j^\star} \quad \text{where} \quad j^\star \,=\, \underset{j \in [k] \setminus \{y\}}{\arg\max} \Big[\log h_j(\mathbf{x}''_j\,; \theta) \,+\, \log h_j(T(\mathbf{x}''_j)\,; \theta) \Big]. \tag{3.14}$$

We also consider the High-Confidence Misclassification Outer Attack (**HCMOA**), which solves for an adversarial input $\mathbf{x}''$ such that the base model has incorrect prediction with high confidence on $\mathbf{x}''$ (details in Appendix A.3).

Final Attack. To get the strongest evaluation, our final attack is an *ensemble of all attacks*, including those designed above and some existing ones. We apply each attack in the ensemble with different hyper-parameters on each clean test input. If any of the attacks achieves the attack goal on an input, then the attack is considered to be successful on it.

3.6 Experiments

In this section, we perform experiments to evaluate the proposed method CPR and compare it with competitive baseline methods. Our main findings are summarized as follows:

1) CPR outperforms the baselines significantly in terms of the total robust loss with respect to different rejection losses, under both seen attacks and unseen attacks;

2) CPR usually has significantly higher robustness with rejection compared to the baselines for small to moderate α under both seen attacks and unseen attacks;

3) CPR also has strong performance under the traditional metrics such as robust accuracy with detection $(1 - R_{\epsilon}^{rej}(f, 0))$.

3.6.1 Experimental Setup

We briefly describe the experimental setup here.

Datasets. We use the MNIST (LeCun, 1998), SVHN (Netzer et al., 2011), and CIFAR-10 (Krizhevsky et al., 2009) datasets. Each dataset has a test set containing 10,000 images. Following Stutz et al. (2020), we compute the accuracy of the models on the first 9,000 images of the test set and compute the robustness of the models on the first 1,000 images of the test set. We use the last 1,000 images of the test set

as a held-out validation set for selecting the hyper-parameters of the methods (e.g., the rejection threshold).

Baselines. We consider the following baselines: (1) AT + CR: adversarial training (AT) (Madry et al., 2018) with Confidence-based Rejection; (2) TRADES + CR: TRADES (Zhang et al., 2019) with Confidence-based Rejection; (3) CCAT (Stutz et al., 2020); (4) RCD (Sheikholeslami et al., 2021); (5) ATRR (Pang et al., 2022).

CPR Setup. CPR requires a base model. We consider robust base models trained using the well-known standard adversarial training (AT) (Madry et al., 2018) and TRADES (Zhang et al., 2019). Our experimental results show that CPR can boost the robustness of these models. CPR has three hyper-parameters: the consistency radius $\tilde{\epsilon}$, the number of PGD steps m, and the PGD step size η. The $\widetilde{\epsilon}$ value controls the rejection rate of the selective classifier. We choose it such that CPR does not reject more than a small fraction of the correctly-classified clean inputs (however, it can reject a majority of the mis-classified inputs that are likely to be close to the decision boundary of the base model; rejecting such inputs is reasonable). The rejection rate of the selective classifier $\mathbf{g}_{\widetilde{\epsilon}}$ on correctly-classified clean inputs from a distribution $\mathcal{D}$ is given by $\mathbb{E}_{(\mathbf{x},y)\sim\mathcal{D}}\left[\mathbf{1}\{\mathbf{g}_{\widetilde{\epsilon}}(\mathbf{x};\theta) = \perp\} \mid \widehat{y}(\mathbf{x}) = y\right]$, which can be estimated using a labeled validation dataset. We choose a large enough $\widetilde{\epsilon} > 0$ such that this rejection rate is approximately p_{rej}, where p_{rej} is a user-specified rejection rate. The number of PGD steps m also affects the robustness. We use a validation set to select suitable $\tilde{\epsilon}$ and m. We do not tune the PGD step size η and set it to a fixed value.

On MNIST, we set $\tilde{\epsilon} = 0.1$ (such that $p_{rej} = 1\%$), $m = 20$, and $\eta = 0.01$. On SVHN and CIFAR-10, we set $\tilde{\epsilon} = 0.0055$ (such that $p_{rej} = 5\%$), $m = 10$, and $\eta = 0.001$.

Evaluation Metrics. We use the *robustness curve* at adversarial budget ϵ (Eq. (3.6)) and the total robust loss to evaluate all the methods. The curve is calculated for α values from the set $\{0, 0.01, 0.05, 0.1, 0.15, 0.2, 0.25, 0.3, 0.5, 1\}$ since in practice, we

Dataset	Method	Total Robust Loss under Seen Attacks ↓					Total Robust Loss under Unseen Attacks ↓				
		Step Rej. Loss			Ramp Rej. Loss		Step Rej. Loss			Ramp Rej. Loss	
		$\alpha_0 = 0.0$	$\alpha_0 = 0.05$	$\alpha_0 = 0.1$	$t = 2$	$t = 4$	$\alpha_0 = 0.0$	$\alpha_0 = 0.05$	$\alpha_0 = 0.1$	$t = 2$	$t = 4$
MNIST	AT+CR	0.084	0.085	0.085	0.097	0.087	1.000	1.000	1.000	1.000	1.000
	TRADES+CR	0.060	0.061	0.061	**0.075**	0.065	1.000	1.000	1.000	1.000	1.000
	CCAT	0.168	0.926	1.000	0.945	0.894	0.245	0.994	1.000	0.955	0.914
	RCD	0.135	0.135	0.135	0.135	0.135	1.000	1.000	1.000	1.000	1.000
	ATRR	0.088	0.088	0.089	0.107	0.095	1.000	1.000	1.000	1.000	1.000
	AT+CPR (Ours)	**0.039**	**0.039**	**0.040**	0.133	0.059	**0.096**	**0.097**	**0.098**	**0.187**	**0.116**
	TRADES+CPR (Ours)	0.042	0.042	0.042	0.130	**0.058**	0.133	0.133	0.133	0.208	0.145
SVHN	AT+CR	0.539	0.539	0.539	0.542	0.540	0.882	0.882	0.882	0.882	0.882
	TRADES+CR	0.471	0.472	0.472	0.475	0.473	0.874	0.874	0.874	0.874	0.874
	CCAT	0.547	1.000	1.000	0.977	0.956	0.945	1.000	1.000	0.999	0.998
	RCD	0.662	0.662	0.662	0.662	0.662	0.903	0.903	0.903	0.903	0.903
	ATRR	0.552	0.552	0.552	0.566	0.556	0.885	0.885	0.885	0.891	0.887
	AT+CPR (Ours)	0.442	0.442	0.442	0.454	0.444	0.853	0.853	0.853	0.856	0.854
	TRADES+CPR (Ours)	**0.380**	**0.380**	**0.380**	**0.390**	**0.381**	**0.813**	**0.813**	**0.813**	**0.816**	**0.814**
CIFAR-10	AT+CR	0.500	0.501	0.501	0.507	0.503	0.895	0.895	0.895	0.895	0.895
	TRADES+CR	0.500	0.500	0.500	0.503	0.501	0.849	0.849	0.849	0.849	0.849
	CCAT	0.723	1.000	1.000	0.984	0.969	0.912	1.000	1.000	0.998	0.997
	RCD	0.533	0.533	0.533	0.533	0.533	0.905	0.905	0.905	0.905	0.905
	ATRR	0.512	0.513	0.513	0.518	0.515	0.887	0.887	0.887	0.887	0.887
	AT+CPR (Ours)	0.433	0.433	0.433	0.443	0.435	0.829	0.829	0.829	0.836	0.830
	TRADES+CPR (Ours)	**0.429**	**0.429**	**0.429**	**0.438**	**0.430**	**0.781**	**0.781**	**0.781**	**0.787**	**0.782**

Table 3.1: The total robust loss for different rejection loss functions under both seen and unseen attacks. The best result is **boldfaced**.

are mainly interested in the robustness with rejection for small values of α. We plot the robustness curve, with α along the x-axis and the robustness with rejection along the y-axis. We also evaluate the *total robust loss* $L_\epsilon(f; \ell^{rej})$ with respect to the ramp rejection loss (Eq. (3.4)) for $t \in \{2, 4\}$, and the step rejection loss (Eq. (3.3)) for $\alpha_0 \in \{0, 0.05, 0.1\}$. This gives a single metric that summarizes the robustness curve of the different methods.

Evaluation. We consider ℓ_∞-norm bounded attacks and generate adversarial examples to compute the robustness with rejection metric via *an ensemble of adversarial attacks*. The worst-case robustness is reported under the attack ensemble. A detailed discussion of the unified approach for designing strong adaptive attacks for CPR and all the baseline methods is given in Appendix A.3. We consider both *seen* attacks and *unseen* attacks. Suppose ϵ' is the attack perturbation budget for testing. For seen attacks, the perturbation budget $\epsilon' = \epsilon$ (on MNIST, $\epsilon' = 0.3$, while on SVHN and CIFAR-10, $\epsilon' = \frac{8}{255}$). For unseen attacks, the perturbation budget $\epsilon' > \epsilon$ (on MNIST, $\epsilon' = 0.4$, while on SVHN and CIFAR-10, $\epsilon' = \frac{16}{255}$). Finally, we use

the same approach to set the rejection threshold for all the baselines. Specifically, on MNIST, we set the threshold such that only 1% of clean correctly-classified validation inputs are rejected. On SVHN and CIFAR-10, we set the threshold such that only 5% of clean correctly-classified validation inputs are rejected.

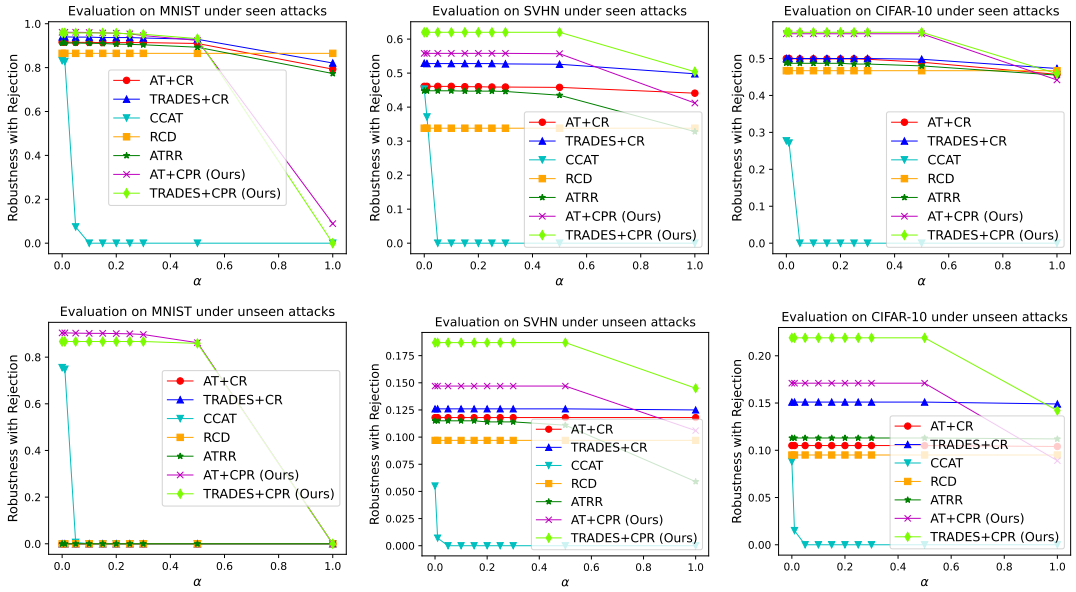


Figure 3.2: The robustness curve of our method CPR and the baselines for both seen and unseen attacks.

Dataset	Method	Robustness with Reject at $\alpha = 0 \downarrow$			Robustness with Reject at $\alpha = 1 \downarrow$		
		Outer Attack			Inner Attack (with Ensemble Outer Attacks)		
		AutoAttack	HCMOA	CHCMOA	LCIA	CLCIA	PDIA
MNIST	AT+CPR	97.60	97.80	**96.10**	50.90	**12.30**	65.80
	TRADES+CPR	98.10	98.40	**95.80**	7.60	5.10	**0.40**
SVHN	AT+CPR	64.20	57.50	**56.70**	**42.60**	49.40	43.90
	TRADES+CPR	71.10	63.40	**62.90**	**51.00**	56.10	54.30
CIFAR-10	AT+CPR	60.70	57.50	**57.40**	**44.40**	48.30	48.50
	TRADES+CPR	61.50	57.40	**57.20**	**45.90**	51.10	53.10

Table 3.2: Ablation study on the outer and inner adaptive attacks for CPR. Refer to Appendix A.3 for more details on these attacks and the choice of metrics. All values are in percentage, and smaller values correspond to a stronger attack. **Bold** values show the strongest attacks.

3.6.2 Results

Evaluating the Total Robust Loss. Table 3.1 compares the total robust loss of different methods for different rejection loss functions under both seen attacks and unseen attacks. The proposed method CPR outperforms the baselines significantly in almost all cases. The only exception is on MNIST for the ramp rejection loss with $t = 2$ under seen attacks, where CPR is worse than TRADES+CR and some baselines. This is because small perturbations on MNIST are easy to correctly classify and the ramp rejection loss penalizes rejecting large perturbations more, while CPR tends to reject large perturbations. In all other cases, CPR performs significantly better than the baselines. For instance, under unseen attacks, CPR reduces the total robust loss (with respect to different rejection losses) by at least 60.8%, 6.6% and 7.3% on MNIST, SVHN and CIFAR-10, respectively. Under seen attacks, CPR reduces the total robust loss by at least 18.0% and 12.8% on SVHN and CIFAR-10, respectively.

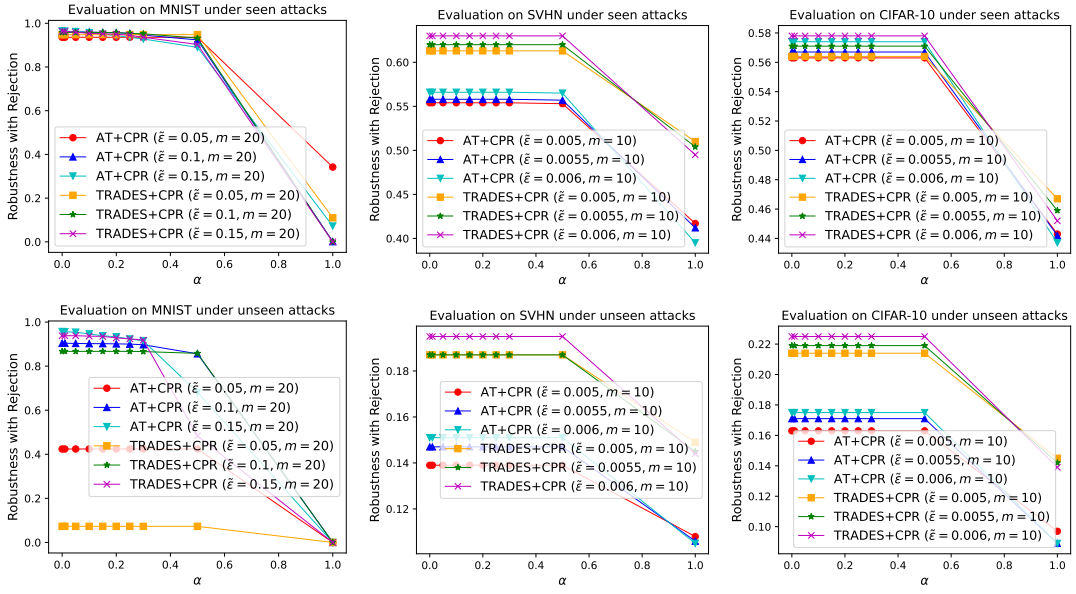


Figure 3.3: Ablation study for the proposed method CPR where we vary the hyper-parameter $\tilde{\epsilon}$ while fixing the hyper-parameter m.

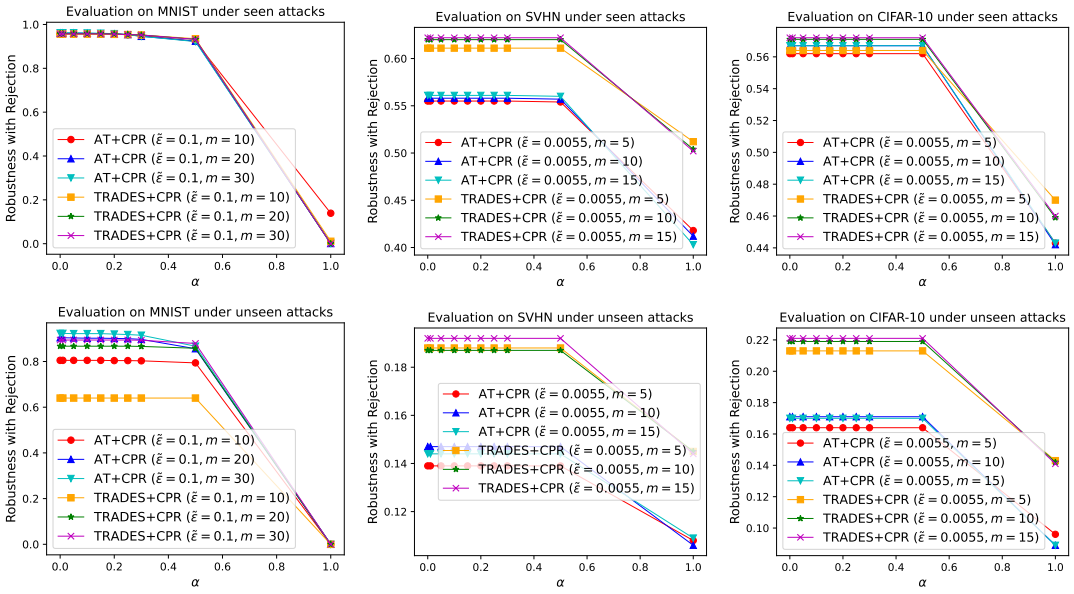


Figure 3.4: Ablation study for the proposed method CPR where we vary the hyper-parameter m while fixing the hyper-parameter $\tilde{\epsilon}$.

Evaluating the Robustness Curve. Figure 3.2 compares the robustness with rejection $A_\epsilon^{\text{rej}}(f, \alpha)$ of the different methods as a function of α under both seen and unseen attacks. Our method CPR usually has significantly higher robustness with rejection compared to the baselines for small to moderate α. The robustness with rejection of CPR only drops for large α values, which suffers less rejection loss and thus leads to smaller total robust loss (as predicted by our analysis). We also note that CCAT has worse performance than other methods since our adaptive attacks are stronger than the PGD-with-backtracking attack used in CCAT paper (Stutz et al., 2020).

Ablation Study on Attacks. We performed an ablation experiment to study the strength of each inner and outer attack in the attack ensemble. The results in Table 3.2 show that for the outer attack, CHCMOA is consistently the best across all datasets. For the inner attack, LCIA is usually the best, and CLCIA and PDIA are strong on MNIST. We emphasize that we use an ensemble of all the attacks to get

the strongest final evaluation.

Ablation Study on Hyper-parameters. We performed an ablation experiment to study the effect of the hyper-parameters $\tilde{\epsilon}$ and m used by CPR. The results in Figure 3.3 and Figure 3.4 show that larger $\tilde{\epsilon}$ (consistency radius) leads to better robustness with rejection at $\alpha = 0$. However, it also leads to lower robustness with rejection when α is large, which suggests that CPR rejects more perturbed inputs. Similarly, larger m (number of PGD steps) also leads to better robustness with rejection at $\alpha = 0$, but can lead to a lower robustness with rejection when α is large. We set $m = 10$ in our main experiments, which is usually sufficient for good performance, and larger values lead to minimal improvements.

Evaluating Traditional Metrics. We also evaluate different methods on the traditional metrics, including accuracy with rejection, rejection rate on clean test inputs, an F1 score-like metric (harmonic mean of accuracy-with-rejection and $1 -$ rejection rate), and the robust accuracy with detection defined in Tramèr (2021). The results in Table 3.3 show that CPR has comparable performance to the baselines on clean test inputs, and also significantly outperforms the baselines w.r.t. robust accuracy with detection.

3.7 Conclusion

In this chapter, we studied adversarially-robust classification with rejection in the practical setting where rejection carried a loss that was monotonically non-increasing with the perturbation magnitude. We proposed the total robust loss as a generalization of the robust error for selective classifiers where rejection carried a loss, and the robustness curve as a tool to study the total robust loss. We provided an analysis of the setting and proposed a novel defense CPR for robustifying any given base model, which significantly outperformed previous methods under strong adaptive attacks.

Dataset	Method	Clean Test Inputs			Under Seen Attacks	Under Unseen Attacks
		Acc. with Rej. ↑	Rej. Rate ↓	F1 Score ↑	Robust Acc. with Det. ↑	Robust Acc. with Det. ↑
MNIST	AT	98.81	0.00	99.40	84.70	0.00
	AT+CR	99.55	1.79	98.87	91.60	0.00
	TRADES	99.07	0.00	**99.53**	89.30	0.00
	TRADES+CR	99.67	1.86	98.90	94.00	0.00
	CCAT	99.90	1.82	99.03	83.20	75.50
	RCD	99.02	0.00	99.51	86.50	0.00
	ATRR	99.62	2.51	98.54	91.20	0.00
	AT+CPR (Ours)	99.60	1.99	98.80	**96.10**	**90.40**
	TRADES+CPR (Ours)	99.63	1.63	98.99	95.80	86.70
SVHN	AT	92.58	0.00	96.15	45.10	11.70
	AT+CR	96.22	8.91	93.58	46.10	11.80
	TRADES	92.19	0.00	95.94	52.00	12.30
	TRADES+CR	95.47	9.06	93.15	52.90	12.60
	CCAT	99.04	7.73	95.53	45.30	5.50
	RCD	96.58	0.00	**98.26**	33.80	9.70
	ATRR	96.14	8.98	93.51	44.80	11.50
	AT+CPR (Ours)	95.86	7.34	94.23	55.80	14.70
	TRADES+CPR (Ours)	94.96	6.56	94.20	**62.00**	**18.70**
CIFAR-10	AT	84.84	0.00	91.80	47.60	10.80
	AT+CR	90.55	13.00	88.74	50.00	10.50
	TRADES	82.12	0.00	90.18	48.70	15.20
	TRADES+CR	86.57	9.59	88.45	50.00	15.10
	CCAT	93.18	9.12	92.01	27.70	8.80
	RCD	88.13	2.07	**92.77**	46.70	9.50
	ATRR	89.36	12.09	88.63	48.80	11.30
	AT+CPR (Ours)	89.05	9.57	89.74	56.70	17.10
	TRADES+CPR (Ours)	86.30	9.57	88.32	**57.10**	**21.90**

Table 3.3: Evaluation of traditional metrics (percentages). Top-1 **boldfaced**.

4 ROBUST ATTRIBUTION REGULARIZATION

Contribution statement. This chapter is joint work with Xi Wu, Vaibhav Rastogi, Yingyu Liang, and Somesh Jha. The author Jiefeng Chen proposed the method, contributed to part of the theoretical analysis, and completed all the experiments. The paper version of this chapter appeared in NeurIPS 2019 (Chen et al., 2019).

4.1 Introduction

Trustworthy machine learning has received considerable attention in recent years. An emerging problem to tackle in this domain is to train models that produce reliable interpretations for their predictions. For example, a pathology prediction model may predict certain images as containing malignant tumor. Then one would hope that under visually indistinguishable perturbations of an image, similar sections of the image, instead of entirely different ones, can account for the prediction. However, as Ghorbani et al. (2019) convincingly demonstrated, for existing models, one can generate minimal perturbations that substantially change model interpretations, *while keeping their predictions intact*. Unfortunately, while the *robust prediction* problem of machine learning models is well known and has been extensively studied in recent years (for example, Madry et al. (2018); Sinha et al. (2018); Wong and Kolter (2018), and also the tutorial by Kolter and Madry (2018)), there has only been limited progress on the problem of *robust interpretations*.

In this chapter, we take a step towards solving this problem by viewing it through the lens of axiomatic attribution of neural networks, and propose Robust Attribution Regularization. Our theory is grounded in the recent work, *Integrated Gradients* (IG) (Sundararajan et al., 2017), in *axiomatically attributing* a neural network's *output change* to *its input change*. Specifically, given a model f, two input vectors $\mathbf{x}, \mathbf{x}'$, and an input coordinate i, $\mathrm{IG}_i^f(\mathbf{x}, \mathbf{x}')$ defines a path integration (parameterized by a curve from $\mathbf{x}$ to $\mathbf{x}'$) that assigns a number to the i-th input as its "contribution" to the change of the model's output from $f(\mathbf{x})$ to $f(\mathbf{x}')$. IG enjoys several natural

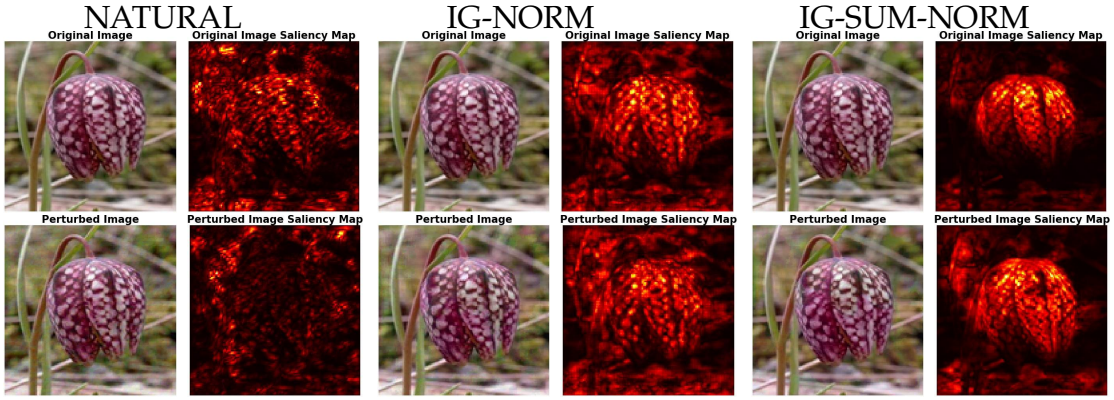


Figure 4.1: **Attribution robustness comparing different models.** Top-1000 Intersection and Kendall's Correlation are rank correlations between original and perturbed saliency maps. NATURAL is the naturally trained model, IG-NORM and IG-SUM-NORM are models trained using our robust attribution method. We use attribution attacks described in Ghorbani et al. (2019) to perturb the attributions while keeping predictions intact. For all images, the models give *correct* prediction – Windflower. However, the saliency maps (also called feature importance maps), computed via IG, show that attributions of the naturally trained model are very fragile, either visually or quantitatively as measured by correlation analyses, while models trained using our method are much more robust in their attributions.

theoretical properties (such as the Axiom of Completeness[1]) that other related methods violate.

We briefly overview our approach. Given a loss function ℓ and a data generating distribution P, our Robust Attribution Regularization objective contains two parts: **(1)** Achieving a small loss over the distribution P, and **(2)** The IG attributions of the loss ℓ over P are "close" to the IG attributions over Q, if distributions P and Q are close to each other. We can naturally encode these two goals in two classic robust optimization models: **(1)** In the *uncertainty set model* (Ben-Tal et al., 2009) where we treat sample points as "nominal" points, and assume that true sample

[1]Axiom of Completeness says that summing up attributions of all components should give $f(\mathbf{x}') - f(\mathbf{x})$.

points are from certain vicinity around them, which gives:

$$\begin{aligned} &\underset{\theta}{\text{minimize}} \quad \underset{(\mathbf{x},y)\sim P}{\mathbb{E}}[\rho(\mathbf{x}, y; \theta)] \\ &\text{where } \ \rho(\mathbf{x}, y; \theta) = \ell(\mathbf{x}, y; \theta) + \lambda \max_{\mathbf{x}' \in N(\mathbf{x}, \varepsilon)} s(IG_{\mathbf{h}}^{\ell_y}(\mathbf{x}, \mathbf{x}'; r)) \end{aligned}$$

where $IG_{\mathbf{h}}^{\ell_y}(\cdot)$ is the attribution w.r.t. neurons in an intermediate layer $\mathbf{h}$, and $s(\cdot)$ is a size function (e.g., $\|\cdot\|_2$) measuring the size of IG, and **(2)** In the *distributional robustness* model (Sinha et al., 2018; Mohajerin Esfahani and Kuhn, 2015), where closeness between P and Q is measured using metrics such as Wasserstein distance, which gives:

$$\underset{\theta}{\text{minimize}} \quad \underset{P}{\mathbb{E}}[\ell(P; \theta)] + \lambda \sup_{Q; M \in \prod(P,Q)} \left\{ \underset{Z,Z'}{\mathbb{E}}[d_{IG}(Z, Z')] \text{ s.t. } \underset{Z,Z'}{\mathbb{E}}[c(Z, Z')] \leqslant \rho \right\},$$

In this formulation, $\prod(P, Q)$ is the set of couplings of P and Q, and $M = (Z, Z')$ is one coupling. $c(\cdot, \cdot)$ is a metric, such as $\|\cdot\|_2$, to measure the cost of an adversary perturbing z to z'. ρ is an upper bound on the expected perturbation cost, thus constraining P and Q to be "close" with each together. d_{IG} is a metric to measure the change of attributions from Z to Z′, where we want a large d_{IG}-change under a small c-change. The supremum is taken over Q and $\prod(P, Q)$.

We provide theoretical characterizations of our objectives. First, we show that they give principled generalizations of previous objectives designed for *robust predictions*. Specifically, under *weak* instantiations of size function $s(\cdot)$, and how we estimate IG computationally, we can leverage axioms satisfied by IG to recover the robust prediction objective of Madry et al. (2018), the input gradient regularization objective of Ross and Doshi-Velez (2018), and also the distributional robust prediction objective of Sinha et al. (2018). These results provide theoretical evidence that robust prediction training can provide some control over robust interpretations. Second, for one-layer neural networks, we prove that instantiating $s(\cdot)$ as 1-norm coincides with the instantiation of $s(\cdot)$ as sum, and further coincides with classic soft-margin training, which implies that for generalized linear classifiers,

soft-margin training will robustify both predictions and interpretations. Finally, we generalize previous theory on distributional robust prediction (Sinha et al., 2018) to our objectives, and show that they are closely related.

Through detailed experiments we study the effect of our method in robustifying attributions. On MNIST, Fashion-MNIST, GTSRB and Flower datasets, we report encouraging improvement in attribution robustness. Compared with naturally trained models, we show significantly improved attribution robustness, as well as prediction robustness. Compared with Madry et al.'s model (Madry et al., 2018) trained for robust predictions, we demonstrate *comparable* prediction robustness (*sometimes even better*), while *consistently* improving attribution robustness. We observe that even when our training stops, the attribution regularization term remains much more significant compared to the natural loss term. We discuss this problem and point out that current optimization techniques may not have effectively optimized our objectives. These results hint at the need for better optimization techniques or new neural network architectures that are more amenable to robust attribution training.

4.2 Preliminaries

Axiomatic attribution and Integrated Gradients Let $f : \mathbb{R}^d \mapsto \mathbb{R}$ be a real-valued function, and $\mathbf{x}$ and $\mathbf{x}'$ be two input vectors. Given that function values changes from $f(\mathbf{x})$ to $f(\mathbf{x}')$, a basic question is: *"How to attribute the function value change to the input variables?"* A recent work by Sundararajan et al. (2017) provides an *axiomatic answer* to this question. Formally, let $r : [0, 1] \mapsto \mathbb{R}^d$ be a curve such that $r(0) = \mathbf{x}$, and $r(1) = \mathbf{x}'$, Integrated Gradients (IG) for input variable i is defined as the following integral:

$$\mathrm{IG}_i^f(\mathbf{x}, \mathbf{x}'; r) = \int_0^1 \frac{\partial f(r(t))}{\partial \mathbf{x}_i} r_i'(t)\, dt, \tag{4.1}$$

which formalizes the contribution of the i-th variable as the integration of the i-th partial as we move along curve r. Let $\mathrm{IG}^f(\mathbf{x}, \mathbf{x}'; r)$ be the vector where the i-th

component is IG^f_i, then IG^f satisfies some natural axioms. For example, the Axiom of Completeness says that summing all coordinates gives the change of function value: $sum(IG^f(\mathbf{x}, \mathbf{x}'; r)) = \sum_{i=1}^{d} IG^f_i(\mathbf{x}, \mathbf{x}'; r) = f(\mathbf{x}') - f(\mathbf{x})$. We refer readers to the paper (Sundararajan et al., 2017) for other axioms IG satisfies.

Integrated Gradients for an intermediate layer. We can generalize the theory of IG to an intermediate layer of neurons. The key insight is to leverage the fact that Integrated Gradients is a *curve integration*. Therefore, given some hidden layer $\mathbf{h} = [h_1, \ldots, h_l]$, computed by a function $h(\mathbf{x})$ induced by previous layers, one can then naturally view the previous layers as inducing a *curve* $h \circ r$ which moves from $h(\mathbf{x})$ to $h(\mathbf{x}')$, as we move from $\mathbf{x}$ to $\mathbf{x}'$ along curve r. Viewed this way, we can thus naturally compute IG for $\mathbf{h}$ in a way that leverages all layers of the network[2],

Lemma 4.1. *Under curve* $r : [0, 1] \mapsto \mathbb{R}^d$ *such that* $r(0) = \mathbf{x}$ *and* $r(1) = \mathbf{x}'$ *for moving* $\mathbf{x}$ *to* $\mathbf{x}'$*, and the function induced by layers before* $\mathbf{h}$*, the attribution for* h_i *for a differentiable* f *is*

$$IG^f_{h_i}(\mathbf{x}, \mathbf{x}') = \sum_{j=1}^{d} \left\{ \int_0^1 \frac{\partial f(h(r(t)))}{\partial h_i} \frac{\partial h_i(r(t))}{\partial \mathbf{x}_j} r'_j(t) dt \right\}. \tag{4.2}$$

The corresponding summation approximation is:

$$IG^f_{h_i}(\mathbf{x}, \mathbf{x}') = \frac{1}{m} \sum_{j=1}^{d} \left\{ \sum_{k=0}^{m-1} \frac{\partial f(h(r(k/m)))}{\partial h_i} \frac{\partial h_i(r(k/m))}{\partial \mathbf{x}_j} r'_j(k/m) \right\} \tag{4.3}$$

4.3 Robust Attribution Regularization

In this section, we propose objectives for achieving robust attribution, and study their connections with existing robust training objectives. At a high level, given a loss function ℓ and a data generating distribution P, our objectives contain two parts: (1) Achieving a small loss over the data generating distribution P, and (2) The IG attributions of the loss ℓ over P are "close" to the IG attributions over distribution

[2]Proofs are deferred to Appendix B.1.2.

Q, if P and Q are close to each other. We can naturally encode these two goals in existing robust optimization models. Below we do so for two popular models: the *uncertainty set model* and the *distributional robustness* model.

4.3.1 Uncertainty Set Model

In the uncertainty set model, for any sample $(\mathbf{x}, y) \sim P$ for a data generating distribution P, we think of it as a "nominal" point and assume that the real sample comes from a neighborhood around $\mathbf{x}$. In this case, given any intermediate layer $\mathbf{h}$, we propose the following objective function:

$$\begin{aligned} &\underset{\theta}{\text{minimize}} \quad \underset{(\mathbf{x},y)\sim P}{\mathbb{E}}[\rho(\mathbf{x}, y; \theta)] \\ &\text{where } \rho(\mathbf{x}, y; \theta) = \ell(\mathbf{x}, y; \theta) + \lambda \max_{\mathbf{x}' \in N(\mathbf{x},\varepsilon)} s(IG^{\ell_y}_{\mathbf{h}}(\mathbf{x}, \mathbf{x}'; r)) \end{aligned} \tag{4.4}$$

where $\lambda \geqslant 0$ is a regularization parameter, ℓ_y is the loss function with label y fixed: $\ell_y(\mathbf{x}; \theta) = \ell(\mathbf{x}, y; \theta)$, $r : [0, 1] \mapsto \mathbb{R}^d$ is a curve parameterization from $\mathbf{x}$ to $\mathbf{x}'$, and IG^{ℓ_y} is the integrated gradients of ℓ_y, and therefore gives attribution of changes of ℓ_y as we go from $\mathbf{x}$ to $\mathbf{x}'$. $s(\cdot)$ is a size function that measures the "size" of the attribution.[3]

We now study some particular instantiations of the objective (4.4). Specifically, we recover existing robust training objectives under *weak* instantiations (such as choosing $s(\cdot)$ as summation function, which is not metric, or use crude approximation of IG), and also derive new instantiations that are natural extensions to existing ones.

Proposition 1 (**Madry et al.'s robust prediction objective**). *If we set* $\lambda = 1$ *, and let* $s(\cdot)$ *be the* sum *function (sum all components of a vector), then for any curve* r *and any intermediate layer* $\mathbf{h}$*, (4.4) is exactly the objective proposed by Madry et al. (2018) where* $\rho(\mathbf{x}, y; \theta) = \max_{\mathbf{x}' \in N(\mathbf{x},\varepsilon)} \ell(\mathbf{x}', y; \theta)$.

[3] We stress that this regularization term depends on model parameters θ through loss function ℓ_y.

We note that: (1) sum is a weak size function which does not give a metric. (2) As a result, while this robust prediction objective falls within our framework, and regularizes robust attributions, it allows a small regularization term where attributions actually change significantly but they cancel each other in summation. Therefore, the control over robust attributions can be weak.

Proposition 2 (**Input gradient regularization**). *For any* $\lambda' > 0$ *and* $q \geqslant 1$, *if we set* $\lambda = \lambda'/\varepsilon^q$, $s(\cdot) = \|\cdot\|_1^q$, *and use only the first term of summation approximation* (4.3) *to approximate* IG, *then* (4.4) *becomes exactly the input gradient regularization of Drucker and LeCun* (1992), *where we have* $\rho(\mathbf{x}, y; \theta) = \ell(\mathbf{x}, y; \theta) + \lambda\|\nabla_{\mathbf{x}}\ell(\mathbf{x}, y; \theta)\|_q^q$.

In the above we have considered instantiations of a weak size function (summation function), which recovers Madry et al.'s objective, and of a weak approximation of IG (picking the first term), which recovers input gradient regularization. In the next example, we pick a nontrivial size function, the 1-norm $\|\cdot\|_1$, use the precise IG, but then we use a *trivial intermediate layer*, the output loss ℓ_y.

Proposition 3 (**Regularizing by attribution of the loss output**). *Let us set* $\lambda = 1$, $s(\cdot) = \|\cdot\|_1$, *and* $\mathbf{h} = \ell_y$ (*the output layer of loss function!*), *then we have* $\rho(\mathbf{x}, y; \theta) = \ell_y(\mathbf{x}) + \max_{\mathbf{x}' \in N(\mathbf{x}, \varepsilon)}\{|\ell_y(\mathbf{x}') - \ell_y(\mathbf{x})|\}$.

We note that this loss function is a "surrogate" loss function for Madry et al.'s loss function because $\ell_y(\mathbf{x}) + \max_{\mathbf{x}' \in N(\mathbf{x}, \varepsilon)}\{|\ell_y(\mathbf{x}') - \ell_y(\mathbf{x})|\} \geqslant \ell_y(\mathbf{x}) + \max_{\mathbf{x}' \in N(\mathbf{x}, \varepsilon)}\{(\ell_y(\mathbf{x}') - \ell_y(\mathbf{x}))\} = \max_{\mathbf{x}' \in N(\mathbf{x}, \varepsilon)} \ell_y(\mathbf{x}')$. Therefore, even at such a trivial instantiation, robust attribution regularization provides interesting guarantees.

4.3.2 Distributional Robustness Model

A different but popular model for robust optimization is the distributional robustness model. In this case we consider a family of distributions $\mathcal{P}$, each of which is supposed to be a "slight variation" of a base distribution P. The goal of robust optimization is then that certain objective functions obtain stable values over this entire family. Here we apply the same underlying idea to the distributional robustness

model: One should get a small loss value over the base distribution P, and for any distribution $Q \in \mathcal{P}$, the IG-based *attributions* change only a little if we move from P to Q. This is formalized as:

$$\underset{\theta}{\text{minimize}} \quad \mathop{\mathbb{E}}_{P}[\ell(P;\theta)] + \lambda \sup_{Q \in \mathcal{P}} \{W_{d_{IG}}(P,Q)\},$$

where the $W_{d_{IG}}(P,Q)$ is the Wasserstein distance between P and Q under a distance metric d_{IG}.[4] We use IG to highlight that this metric is related to integrated gradients.

We propose again $d_{IG}(\mathbf{z},\mathbf{z}') = s(IG^{\ell}_{\mathbf{h}}(\mathbf{z},\mathbf{z}'))$. We are particularly interested in the case where $\mathcal{P}$ is a Wasserstein ball around the base distribution P, using "perturbation" cost metric $c(\cdot)$. This gives regularization term $\lambda\, \mathbb{E}_{W_c(P,Q)\leqslant\rho} \sup\{W_{d_{IG}}(P,Q)\}$. An unsatisfying aspect of this objective, as one can observe now, is that $W_{d_{IG}}$ and W_c can take two *different* couplings, while intuitively we want to use only one coupling to transport P to Q. For example, this objective allows us to pick a coupling M_1 under which we achieve $W_{d_{IG}}$ (recall that Wasserstein distance is an infimum over couplings), and a different coupling M_2 under which we achieve W_c, but under $M_1 = (Z, Z')$, $\mathbb{E}_{z,z'\sim M_1}[c(z,z')] > \rho$, violating the constraint. This motivates the following modification:

$$\underset{\theta}{\text{minimize}} \quad \mathop{\mathbb{E}}_{P}[\ell(P;\theta)] + \lambda \sup_{Q;M\in\prod(P,Q)} \Big\{ \mathop{\mathbb{E}}_{Z,Z'}[d_{IG}(Z,Z')] \text{ s.t. } \mathop{\mathbb{E}}_{Z,Z'}[c(Z,Z')] \leqslant \rho \Big\}, \tag{4.5}$$

In this formulation, $\prod(P,Q)$ is the set of couplings of P and Q, and $M = (Z, Z')$ is one coupling. $c(\cdot,\cdot)$ is a metric, such as $\|\cdot\|_2$, to measure the cost of an adversary perturbing z to z'. ρ is an upper bound on the expected perturbation cost, thus constraining P and Q to be "close" with each together. d_{IG} is a metric to measure the change of attributions from Z to Z', where we want a large d_{IG}-change under a small c-change. The supremum is taken over Q and $\prod(P,Q)$.

[4] For supervised learning problem where P is of the form $Z = (X, Y)$, we use the same treatment as in Sinha et al. (2018) so that cost function is defined as $c(z,z') = c_x(x,x') + \infty \cdot \mathbf{1}\{y \neq y'\}$. All our theory carries over to such c which has range $\mathbb{R}_+ \cup \{\infty\}$.

Proposition 4 (**Wasserstein prediction robustness**). *Let* $s(\cdot)$ *be the summation function and* $\lambda = 1$*, then for any curve* γ *and any layer* $\mathbf{h}$*,* (4.5) *reduces to*

$$\sup_{Q:W_c(P,Q)\leqslant\rho} \left\{ \mathop{\mathbb{E}}_{Q}[\ell(Q;\theta)] \right\}$$

, which is the objective proposed by Sinha et al. (*2018*) *for* robust predictions.

Lagrange relaxation. For any $\gamma \geqslant 0$, the Lagrange relaxation of (4.5) is

$$\underset{\theta}{\text{minimize}} \quad \left\{ \mathop{\mathbb{E}}_{P}[\ell(P;\theta)] + \lambda \sup_{Q;M\in\prod(P,Q)} \left\{ \mathop{\mathbb{E}}_{M=(Z,Z')} \left[d_{IG}(Z,Z') - \gamma c(Z,Z') \right] \right\} \right\} \tag{4.6}$$

where the supremum is taken over Q (unconstrained) and all couplings of P and Q, and we want to find a coupling under which IG attributions change a lot, while the perturbation cost from P to Q with respect to c is small. Recall that $g : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ is a *normal integrand* if for each α, the mapping $z \to \{z' | g(z,z') \leqslant \alpha\}$ is closed-valued and measurable (Rockafellar and Wets, 2009).

Our next two theorems generalize the duality theory in Sinha et al. (2018) to a much larger, but natural, class of objectives.

Theorem 4.2. *Suppose* $c(z,z) = 0$ *and* $d_{IG}(z,z) = 0$ *for any* z*, and suppose* $\gamma c(z,z') - d_{IG}(z,z')$ *is a normal integrand. Then,*

$$\sup_{Q;M\in\prod(P,Q)} \{ \mathop{\mathbb{E}}_{M=(Z,Z')} [d_{IG}^{\gamma}(Z,Z')] \} = \mathop{\mathbb{E}}_{z\sim P} [\sup_{z'} \{ d_{IG}^{\gamma}(z,z') \}].$$

Consequently, we have (*4.6*) *to be equal to the following:*

$$\underset{\theta}{\text{minimize}} \quad \mathop{\mathbb{E}}_{z\sim P} \left[\ell(z;\theta) + \lambda \sup_{z'} \{ d_{IG}(z,z') - \gamma c(z,z') \} \right] \tag{4.7}$$

The assumption $d_{IG}(z,z) = 0$ is true for what we propose, and $c(z,z) = 0$ is true for any typical cost such as ℓ_p distances. The normal integrand assumption is also very weak, e.g., it is satisfied when d_{IG} is continuous and c is closed convex.

Note that (4.7) and (4.4) are very similar, and so we use (4.4) for the rest of the chapter. Finally, given Theorem 4.2, we are also able to connect (4.5) and (4.7) with the following duality result:

Theorem 4.3. *Suppose* $c(z,z) = 0$ *and* $d_{IG}(z,z) = 0$ *for any* z*, and suppose* $\gamma c(z,z') - d_{IG}(z,z')$ *is a normal integrand. For any* $\rho > 0$*, there exists* $\gamma \geqslant 0$ *such that the optimal solutions of* (4.7) *are optimal for* (4.5)*.*

4.3.3 One Layer Neural Networks

We now consider the special case of one-layer neural networks, where the loss function takes the form of $\ell(\mathbf{x}, y; \mathbf{w}) = g(-y\langle \mathbf{w}, \mathbf{x}\rangle)$, $\mathbf{w}$ is the model parameters, $\mathbf{x}$ is a feature vector, y is a label, and g is nonnegative. We take $s(\cdot)$ to be $\|\cdot\|_1$, which corresponds to a strong instantiation that does not allow attributions to cancel each other. Interestingly, we prove that for natural choices of g, this is however exactly Madry et al.'s objective (Madry et al., 2018), which corresponds to $s(\cdot) = sum(\cdot)$. That is, the strong ($s(\cdot) = \|\cdot\|_1$) and weak instantiations ($s(\cdot) = sum(\cdot)$) coincide for one-layer neural networks. This thus says that for generalized linear classifiers, "robust interpretation" coincides with "robust predictions," and further with classic soft-margin training.

Theorem 4.4. *Suppose that* g *is differentiable, non-decreasing, and convex. Then for* $\lambda = 1$*,* $s(\cdot) = \|\cdot\|_1$*, and* ℓ_∞ *neighborhood,* (4.4) *reduces to Madry et al.'s objective:*

$$\begin{aligned}&\sum_{i=1}^{m} \max_{\|x_i' - x_i\|_\infty \leqslant \varepsilon} g(-y_i\langle \mathbf{w}, \mathbf{x}_i'\rangle) \quad (\textit{Madry et al.'s objective})\\ =&\sum_{i=1}^{m} g(-y_i\langle \mathbf{w}, \mathbf{x}_i\rangle + \varepsilon\|\mathbf{w}\|_1) \quad (\textit{soft-margin}).\end{aligned}$$

Natural losses, such as Negative Log-Likelihood and softplus hinge loss, satisfy the conditions of this theorem.

4.4 Instantiations and Optimizations

In this section we discuss instantiations of (4.4) and how to optimize them. We start by presenting two objectives instantiated from our method: (1) IG-NORM, and (2) IG-SUM-NORM. Then we discuss how to use gradient descent to optimize these objectives.

IG-NORM. As our first instantiation, we pick $s(\cdot) = \|\cdot\|_1$, $\mathbf{h}$ to be the input layer, and r to be the straightline connecting $\mathbf{x}$ and $\mathbf{x}'$. This gives:

$$\underset{\theta}{\text{minimize}} \quad \mathop{\mathbb{E}}_{(\mathbf{x},y)\sim P}\left[\ell(\mathbf{x},y;\theta) + \lambda \max_{\mathbf{x}'\in N(\mathbf{x},\varepsilon)} \|\operatorname{IG}^{\ell_y}(\mathbf{x},\mathbf{x}')\|_1\right]$$

IG-SUM-NORM. In the second instantiation we combine the sum size function and norm size function, and define $s(\cdot) = \operatorname{sum}(\cdot) + \beta\|\cdot\|_1$. Where $\beta \geqslant 0$ is a regularization parameter. Now with the same $\mathbf{h}$ and r as above, and put $\lambda = 1$, then our method simplifies to:

$$\underset{\theta}{\text{minimize}} \quad \mathop{\mathbb{E}}_{(\mathbf{x},y)\sim P}\left[\max_{\mathbf{x}'\in N(\mathbf{x},\varepsilon)} \left\{\ell(\mathbf{x}',y;\theta) + \beta\|\operatorname{IG}^{\ell_y}(\mathbf{x},\mathbf{x}')\|_1\right\}\right]$$

which can be viewed as appending an extra robust IG term to $\ell(\mathbf{x}')$.

Gradient descent optimization. We propose the following gradient descent framework to optimize the objectives. The framework is parameterized by an adversary $\mathcal{A}$ which is supposed to solve the inner max by finding a point $\mathbf{x}^\star$ which changes attribution significantly. Specifically, given a point $(\mathbf{x}, y)$ at time step t during SGD training, we have the following two steps (this can be easily generalized to mini-batches):

Attack step. We run $\mathcal{A}$ on $(\mathbf{x}, y)$ to find $\mathbf{x}^\star$ that produces a large inner max term (that is $\|\operatorname{IG}^{\ell_y}(\mathbf{x},\mathbf{x}^\star)\|_1$ for IG-NORM, and $\ell(\mathbf{x}^\star) + \beta\|\operatorname{IG}^{\ell_y}(\mathbf{x},\mathbf{x}^\star)\|_1$ for IG-SUM-NORM.

Gradient step. Fixing $\mathbf{x}^\star$, we can then compute the gradient of the corresponding objective with respect to θ, and then update the model.

Important objective parameters. In both attack and gradient steps, we need to differentiate IG (in attack step, θ is fixed and we differentiate w.r.t. $\mathbf{x}$, while in gradient

step, this is reversed), and this induces a set of parameters of the objectives to tune for optimization, which is summarized in Table 4.1. Differentiating summation approximation of IG amounts to compute second partial derivatives. We rely on the auto-differentiation capability of TensorFlow (Abadi et al., 2016) to compute second derivatives.

Adversary $\mathcal{A}$	Adversary to find $\mathbf{x}^\star$. Note that our goal is simply to maximize the inner term in a neighborhood, thus in this chapter, we choose Projected Gradient Descent for this purpose.
m in the attack step	To differentiate IG in the attack step, we use summation approximation of IG, and this is the number of segments for apprioximation.
m in the gradient step	Same as above, but in the gradient step. We have this m separately due to efficiency consideration.
λ	Regularization parameter for IG-NORM.
β	Regularization parameter for IG-SUM-NORM.

Table 4.1: Optimization parameters.

4.5 Experiments

We now perform experiments using our method. We ask the following questions: **(1)** Comparing models trained by our method and naturally trained models at *test time*, do we maintain the accuracy on unperturbed test inputs? **(2)** At test time, if we use attribution attacks mentioned in Ghorbani et al. (2019) to perturb attributions while keeping predictions intact, how does the attribution robustness of our models compare with that of the naturally trained models? **(3)** Finally, how do we compare attribution robustness of our models with *weak instantiations* for robust predictions?

To answer these questions, We perform experiments on four classic datasets: MNIST (LeCun, 1998), Fashion-MNIST (Xiao et al., 2017), GTSRB (Stallkamp et al., 2012), and Flower (Nilsback and Zisserman, 2006). In summary, our findings

are the following: (**1**) Our method results in very small drop in test accuracy compared with naturally trained models. (**2**) On the other hand, our method gives signficantly better attribution robustness, as measured by correlation analyses. (**3**) Finally, our models yield *comparable* prediction robustness (sometimes even better), while *consistently* improving attribution robustness. In the rest of the section we give more details.

Evaluation setup. In this chapter, we use IG to compute attributions (i.e. feature importance map), which, as demonstrated by Ghorbani et al. (2019), is more robust compared to other related methods (note that, IG also enjoys other theoretical properties). To attack attribution while retaining model predictions, we use Iterative Feature Importance Attacks (IFIA) proposed by Ghorbani et al. (2019). We use two metrics to measure attribution robustness (i.e. how similar the attributions are between original and perturbed images):

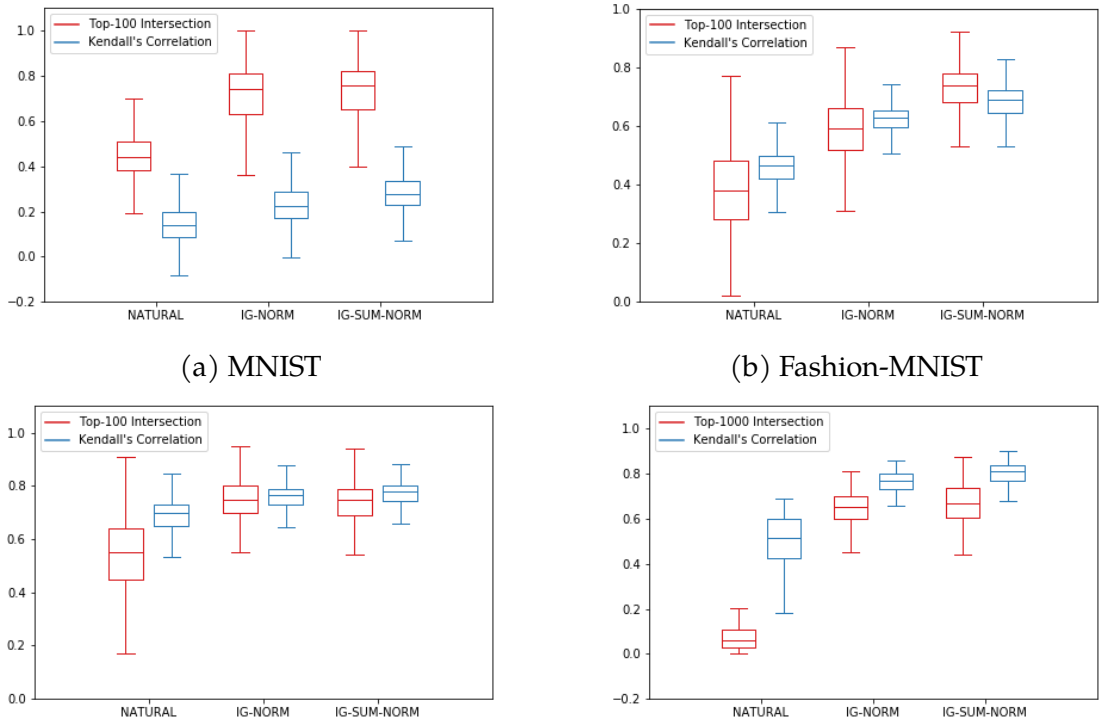
Kendall's tau rank order correlation. Attribution methods rank all of the features in order of their importance, we thus use the rank correlation (Kendall, 1938) to compare similarity between interpretations.

Top-k intersection. We compute the size of intersection of the k most important features before and after perturbation.

Compared with Ghorbani et al. (2019), we use Kendall's tau correlation, instead of Spearman's rank correlation. The reason is that we found that on the GTSRB and Flower datasets, Spearman's correlation is not consistent with visual inspection, and often produces too high correlations. In comparison, Kendall's tau correlation consistently produces lower correlations and aligns better with visual inspection. Finally, when computing attribution robustness, we only consider the test samples that are correctly classified by the model.

Comparing with natural models. Figures (a), (b), (c), and (d) in Figure 4.2 show that, compared with naturally trained models, robust attribution training gives significant improvements in attribution robustness (measured by either median or confidence intervals). The exact numbers are recorded in Table 4.2: Compared with naturally trained models (rows where "Approach" is NATURAL), robust attribution training has significantly better adversarial accuracy and attribution

robustness, while having a small drop in natural accuracy (denoted by `Nat Acc.`).



(a) MNIST (b) Fashion-MNIST

(c) GTSRB (d) Flower

Figure 4.2: Experiment results on MNIST, Fashion-MNIST, GTSRB and Flower.

Ineffective optimization. We observe that even when our training stops, the attribution regularization term remains much more significant compared to the natural loss term. For example for IG-NORM, when training stops on MNIST, $\ell(\mathbf{x})$ typically stays at 1, but $\|\operatorname{IG}(\mathbf{x}, \mathbf{x}')\|_1$ stays at $10 \sim 20$. This indicates that optimization has not been very effective in minimizing the regularization term. There are two possible reasons to this: (1) Because we use summation approximation of IG, it forces us to compute second derivatives, which may not be numerically stable for deep networks. (2) The network architecture may be inherently unsuitable for robust attributions, rendering the optimization hard to converge.

Comparing with robust prediction models. Finally we compare with Madry et al.'s models, which are trained for robust prediction. We use `Adv Acc.` to denote ad-

versarial accuracy (prediction accuracy on perturbed inputs). Again, `TopK Inter.` denotes the average topK intersection (K = 100 for MNIST, Fashion-MNIST and GTSRB datasets, K = 1000 for Flower), and `Rank Corr.` denotes the average Kendall's rank order correlation. Table 4.2 gives the details of the results. As we can see, our models give comparable adversarial accuracy, and are sometimes even better (on the Flower dataset). On the other hand, we are consistently better in terms of attribution robustness.

Dataset	Approach	`Nat Acc.`	`Adv Acc.`	`TopK Inter.`	`Rank Corr.`
MNIST	NATURAL	99.17%	0.00%	46.61%	0.1758
	Madry et al.	98.40%	92.47%	62.56%	0.2422
	IG-NORM	98.74%	81.43%	71.36%	0.2841
	IG-SUM-NORM	98.34%	88.17%	**72.45%**	**0.3111**
Fashion-MNIST	NATURAL	90.86%	0.01%	39.01%	0.4610
	Madry et al.	85.73%	73.01%	46.12%	0.6251
	IG-NORM	85.13%	65.95%	59.22%	0.6171
	IG-SUM-NORM	85.44%	70.26%	**72.08%**	**0.6747**
GTSRB	NATURAL	98.57%	21.05%	54.16%	0.6790
	Madry et al.	97.59%	83.24%	68.85%	0.7520
	IG-NORM	97.02%	75.24%	**74.81%**	0.7555
	IG-SUM-NORM	95.68%	77.12%	74.04%	**0.7684**
Flower	NATURAL	86.76%	0.00%	8.12%	0.4978
	Madry et al.	83.82%	41.91%	55.87%	0.7784
	IG-NORM	85.29%	24.26%	64.68%	0.7591
	IG-SUM-NORM	82.35%	47.06%	**66.33%**	**0.7974**

Table 4.2: Experiment results including prediction accuracy, prediction robustness and attribution robustness.

4.6 Discussion and Conclusion

In this chapter, we built a theory to robustify model interpretations through the lens of axiomatic attributions of neural networks. We showed that our theory gave principled generalizations of previous formulations for robust predictions, and we characterized our objectives for one-layer neural networks.

We believe that this chapter opens many intriguing avenues for future research, and we discuss a few topics below.

Why we want robust attributions? Model attributions are *facts* about model behaviors. While robust attribution does not necessarily mean that the attribution is correct, a model with *brittle attribution* can never be trusted. To this end, it seems interesting to examine attribution methods other than Integrated Gradients.

Robust attribution leads to more human-aligned attribution. Note that our proposed training scheme requires both prediction correctness and robust attributions, and therefore it encourages to learn *invariant* features from data that are also highly predictive. In our experiments, we found an intriguing phenomenon that *our regularized models produce attributions that are much more aligned with human perceptions* (for example, see Figure 4.1). Our results are aligned with the recent work (Tsipras et al., 2019; Engstrom et al., 2019).

Robust attribution may help tackle spurious correlations. In view of our discussion so far, we think it is plausible that robust attribution regularization can help remove spurious correlations because intuitively spurious correlations should not be able to be reliably attributed to. Future research on this potential connection seems warranted.

Difficulty of optimization. While our experimental results are encouraging, we observe that when training stops, the attribution regularization term remains significant (typically around tens to hundreds), which indicates ineffective optimization for the objectives. To this end, a main problem is network depth, where as depth increases, we get very unstable trajectories of gradient descent, which seems to be related to the use of *second order information* during robust attribution optimization (due to summation approximation, we have first order terms in the training objectives). Therefore, it is natural to further study better optimization techniques or better architectures for robust attribution training.

5 ATOM: ROBUSTIFYING OUT-OF-DISTRIBUTION DETECTION USING OUTLIER MINING

Contribution statement. This chapter is joint work with Yixuan Li, Xi Wu, Yingyu Liang, and Somesh Jha. The author Jiefeng Chen proposed the method, contributed to part of the theoretical analysis, and completed all the experiments. The paper version of this chapter appeared in ECML 2021 (Chen et al., 2021a).

5.1 Introduction

Out-of-distribution (OOD) detection has become an indispensable part of building reliable open-world machine learning models (Bendale and Boult, 2015). An OOD detector determines whether an input is from the same distribution as the training data, or different distribution. As of recently a plethora of exciting literature has emerged to combat the problem of OOD detection (Hein et al., 2019; Hsu et al., 2020; Huang and Li, 2021; Lakshminarayanan et al., 2017; Lee et al., 2018b; Liang et al., 2018; Lin et al., 2021; Liu et al., 2020; Mohseni et al., 2020).

Despite the promise, previous methods primarily focused on clean OOD data, while largely underlooking the robustness aspect of OOD detection. Concerningly, recent works have shown the brittleness of OOD detection methods under adversarial perturbations (Bitterwolf et al., 2020; Hein et al., 2019; Sehwag et al., 2019). As illustrated in Figure 5.1, an OOD image (*e.g.*, mailbox) can be perturbed to be misclassified by the OOD detector as in-distribution (traffic sign data). Failing to detect such an *adversarial OOD example*[1] can be consequential in safety-critical applications such as autonomous driving (Filos et al., 2020). Empirically on CIFAR-10, our analysis reveals that the false positive rate (FPR) of a competitive method Outlier Exposure (Hendrycks et al., 2019b) can increase from 3.66% to 99.94% under adversarial attack.

[1]Adversarial OOD examples are constructed w.r.t the OOD detector, which is different from the standard notion of adversarial examples (constructed w.r.t the classification model).

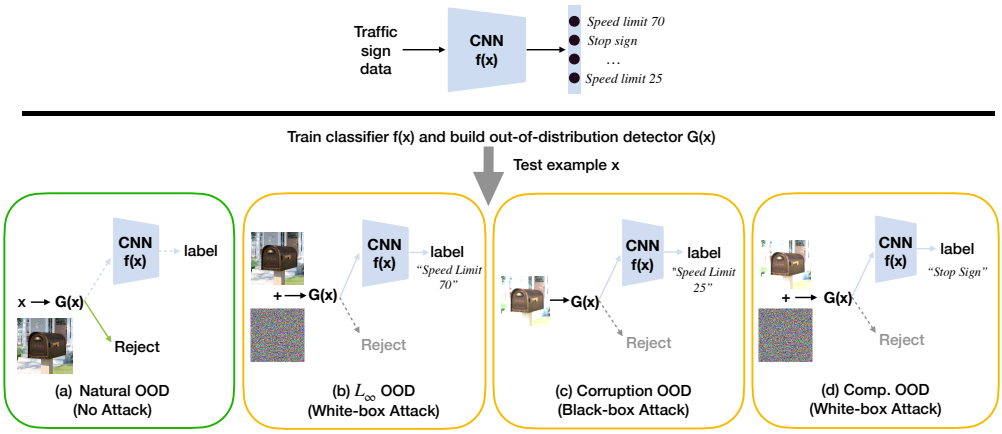


Figure 5.1: **Robust out-of-distribution detection**. When deploying an image classification system (OOD detector $G(x)$ + image classifier $f(x)$) in an open world, there can be multiple types of OOD examples. We consider a broad family of OOD inputs, including (a) Natural OOD, (b) L_∞ OOD, (c) corruption OOD, and (d) Compositional OOD. A detailed description of these OOD inputs can be found in Section 5.4.1. In (b-d), a perturbed OOD input (e.g., a perturbed mailbox image) can mislead the OOD detector to classify it as an in-distribution sample. This can trigger the downstream image classifier $f(x)$ to predict it as one of the in-distribution classes (e.g., speed limit 70). Through *adversarial training with informative outlier mining* (ATOM), our method can robustify the decision boundary of OOD detector $G(x)$, which leads to improved performance across all types of OOD inputs. Solid lines are actual computation flow.

Motivated by this, we make an important step towards the robust OOD detection problem, and propose a novel training framework, ***A**dversarial **T**raining with informative **O**utlier **M**ining* (ATOM). Our key idea is to *selectively* utilize auxiliary outlier data for estimating a tight decision boundary between ID and OOD data, which leads to robust OOD detection performance. While recent methods (Hein et al., 2019; Hendrycks et al., 2019b; Meinke and Hein, 2020; Mohseni et al., 2020) have leveraged auxiliary OOD data, we show that *randomly* selecting outlier samples for training yields a large portion of uninformative samples, which do not meaningfully improve the decision boundary between ID and OOD data (see Figure 5.2). In this chapter, we demonstrate that by mining low OOD score data for training, one can significantly improve the robustness of an OOD detector, and somewhat surprisingly, generalize to unseen adversarial attacks.

We extensively evaluate ATOM on common OOD detection benchmarks, as well as a suite of adversarial OOD tasks, as illustrated in Figure 5.1. ATOM achieves state-of-the-art performance, significantly outperforming competitive methods using standard training on random outliers (Hendrycks et al., 2019b; Meinke and Hein, 2020; Mohseni et al., 2020), or using adversarial training on random outlier data (Hein et al., 2019). On the classic OOD evaluation task (clean OOD data), ATOM achieves comparable and often better performance than current state-of-the-art methods. On L_∞ OOD evaluation task, ATOM outperforms the best baseline ACET (Hein et al., 2019) by a large margin (e.g. **53.9%** false positive rate deduction on CIFAR-10). Moreover, our ablation study underlines the importance of having both adversarial training and outlier mining (ATOM) for achieving robust OOD detection.

Lastly, we provide theoretical analysis for ATOM, characterizing how outlier mining can better shape the decision boundary of the OOD detector. While hard negative mining has been explored in different domains of learning, e.g., object detection, deep metric learning (Felzenszwalb et al., 2009; Gidaris and Komodakis, 2015; Shrivastava et al., 2016), the vast literature of OOD detection has not explored this idea. Moreover, most uses of hard negative mining are on a heuristic basis, but in this chapter, we derive precise formal guarantees with insights. Our **key contributions** are summarized as follows:

- We propose a novel training framework, adversarial training with outlier mining (ATOM), which facilitates efficient use of auxiliary outlier data to regularize the model for robust OOD detection.
- We perform extensive analysis and comparison with a diverse collection of OOD detection methods using: (1) pre-trained models, (2) models trained on randomly sampled outliers, (3) adversarial training. ATOM establishes **state-of-the-art** performance under a broad family of clean and adversarial OOD evaluation tasks.
- We contribute theoretical analysis formalizing the intuition of mining informative outliers for improving the robustness of OOD detection.

- Lastly, we provide a unified evaluation framework that allows future research examining the robustness of OOD detection algorithms under a broad family of OOD inputs.

5.2 Preliminaries

We consider the setting of multi-class classification. We consider a training dataset $\mathcal{D}_{\text{in}}^{\text{train}}$ drawn i.i.d. from a data distribution $P_{\mathbf{X},Y}$, where $\mathbf{X}$ is the sample space and $Y = \{1, 2, \cdots, K\}$ is the set of labels. In addition, we have an auxiliary outlier data $\mathcal{D}_{\text{out}}^{\text{auxiliary}}$ from distribution $U_{\mathbf{X}}$. The use of auxiliary outliers helps regularize the model for OOD detection, as shown in several recent works (Hein et al., 2019; Lee et al., 2018a; Liu et al., 2020; Meinke and Hein, 2020; Mohseni et al., 2020).

Robust out-of-distribution detection. The goal is to learn a detector $G : x \to \{-1, 1\}$, which outputs 1 for an in-distribution example x and output -1 for a clean or perturbed OOD example x. Formally, let $\Omega(x)$ be a set of small perturbations on an OOD example x. The detector is evaluated on x from $P_{\mathbf{X}}$ and on the worst-case input inside $\Omega(x)$ for an OOD example x from $Q_{\mathbf{X}}$. The false negative rate (FNR) and false positive rate (FPR) are defined as:

$$\text{FNR}(G) = \mathbb{E}_{x \sim P_{\mathbf{X}}} \mathbb{I}[G(x) = -1], \quad \text{FPR}(G; Q_{\mathbf{X}}, \Omega) = \mathbb{E}_{x \sim Q_{\mathbf{X}}} \max_{\delta \in \Omega(x)} \mathbb{I}[G(x + \delta) = 1].$$

Remark. Note that test-time OOD distribution $Q_{\mathbf{X}}$ is unknown, which can be different from $U_{\mathbf{X}}$. The difference between the auxiliary data $U_{\mathbf{X}}$ and test OOD data $Q_{\mathbf{X}}$ raises the fundamental question of how to effectively leverage $\mathcal{D}_{\text{out}}^{\text{auxiliary}}$ for improving learning the decision boundary between in- vs. OOD data. For terminology clarity, we refer to training OOD examples as *outliers*, and exclusively use *OOD data* to refer to test-time anomalous inputs.

5.3 Method

In this section, we introduce *Adversarial Training with informative Outlier Mining* (ATOM). We first present our method overview, and then describe details of the training objective with informative outlier mining.

Method overview: a conceptual example. We use the terminology *outlier mining* to denote the process of selecting informative outlier training samples from the pool of auxiliary outlier data. We illustrate our idea with a toy example in Figure 5.2, where in-distribution data consists of class-conditional Gaussians. Outlier training data is sampled from a uniform distribution from outside the support of in-distribution. Without outlier mining (*left*), we will almost sample those "easy" outliers and the decision boundary of the OOD detector learned can be loose. In contrast, with outlier mining (*right*), selective outliers close to the decision boundary between ID and OOD data, which improves OOD detection. This is particularly important for robust OOD detection where the boundary needs to have a margin from the OOD data so that even adversarial perturbation (red color) cannot move the OOD data points across the boundary. We proceed with describing the training mechanism that achieves our novel conceptual idea and will provide formal theoretical guarantees in Section 5.5.

5.3.1 ATOM: Adversarial Training with Informative Outlier Mining

Training objective. The classification involves using a mixture of ID data and outlier samples. Specifically, we consider a $(K+1)$-way classifier network f, where the $(K+1)$-th class label indicates out-of-distribution class. Denote by $F_\theta(x)$ the softmax output of f on x. The robust training objective is given by

$$\underset{\theta}{\text{minimize}} \quad \mathbb{E}_{(x,y)\sim\mathcal{D}_{\text{in}}^{\text{train}}}[\ell(x,y;F_\theta)] + \lambda\cdot\mathbb{E}_{x\sim\mathcal{D}_{\text{out}}^{\text{train}}}\max_{x'\in\Omega_{\infty,\epsilon}(x)}[\ell(x',K+1;F_\theta)] \quad (5.1)$$

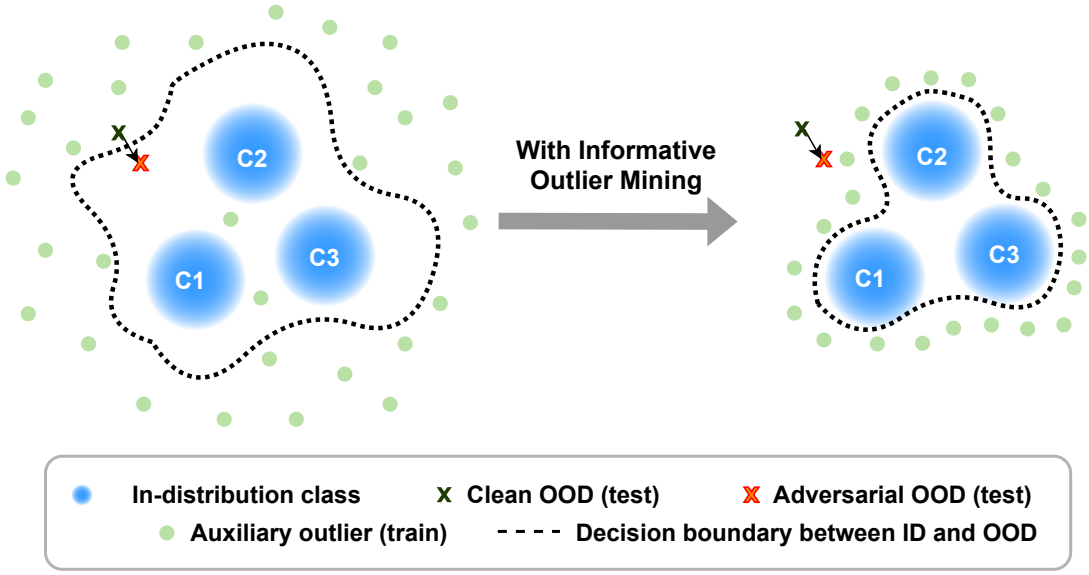


Figure 5.2: A toy example in 2D space for illustration of informative outlier mining. With informative outlier mining, we can tighten the decision boundary and build a robust OOD detector.

where ℓ is the cross entropy loss, and $\mathcal{D}_{\text{out}}^{\text{train}}$ is the OOD training dataset. We use Projected Gradient Descent (PGD) (Madry et al., 2018) to solve the inner max of the objective, and apply it to half of a minibatch while keeping the other half clean to ensure performance on both clean and perturbed data.

Once trained, the OOD detector $G(x)$ can be constructed by:

$$G(x) = \begin{cases} -1 & \text{if } F(x)_{K+1} \geqslant \gamma, \\ 1 & \text{if } F(x)_{K+1} < \gamma, \end{cases} \tag{5.2}$$

where γ is the threshold, and in practice can be chosen on the in-distribution data so that a high fraction of the test examples are correctly classified by G. We call $F(x)_{K+1}$ the *OOD score* of x. For an input labeled as in-distribution by G, one can

obtain its semantic label using $\hat{F}(x)$:

$$\hat{F}(x) = \underset{y \in \{1,2,\cdots,K\}}{\arg\max}\ F(x)_y \tag{5.3}$$

Informative outlier mining. We propose to adaptively choose OOD training examples where the detector is uncertain about. Specifically, during each training epoch, we randomly sample N data points from the auxiliary OOD dataset $\mathcal{D}_{\text{out}}^{\text{auxiliary}}$, and use the current model to infer the OOD scores[2]. Next, we sort the data points according to the OOD scores and select a subset of $n < N$ data points, starting with the qN^{th} data in the sorted list. We then use the selected samples as OOD training data $\mathcal{D}_{\text{out}}^{\text{train}}$ for the next epoch of training. Intuitively, q determines the *informativeness* of the sampled points w.r.t the OOD detector. The larger q is, the less informative those sampled examples become. Note that informative outlier mining is performed on (non-adversarial) auxiliary OOD data. Selected examples are then used in the robust training objective (5.1).

Algorithm 4 ATOM: Adversarial Training with informative Outlier Mining

Require: $\mathcal{D}_{\text{in}}^{\text{train}}$, $\mathcal{D}_{\text{out}}^{\text{auxiliary}}$, F_θ, m, N, n, q
  for $t = 1, 2, \cdots, m$ **do**
    Randomly sample N data points from $\mathcal{D}_{\text{out}}^{\text{auxiliary}}$ to get a candidate set $\mathcal{S}$;
    Compute OOD scores on $\mathcal{S}$ using F_θ to get set $V = \{F(x)_{K+1} \mid x \in \mathcal{S}\}$;
    Sort scores in V from the lowest to the highest;
    $\mathcal{D}_{\text{out}}^{\text{train}} \leftarrow V[qN : qN + n]$; $\{q \in [0, 1 - n/N]\}$
    Train F_θ for one epoch using the training objective of (5.1);
  end for
  Build G and $\hat{F}$ using (5.2) and (5.3) respectively;
  return $\hat{F}$, G

We provide the complete training algorithm using informative outlier mining in Algorithm 4. Importantly, the use of informative outlier mining highlights the key difference between ATOM and previous work using ***randomly*** sampled

[2]Since the inference stage can be fully parallel, outlier mining can be applied with relatively low overhead.

outliers (Hein et al., 2019; Hendrycks et al., 2019b; Meinke and Hein, 2020; Mohseni et al., 2020).

5.4 Experiments

In this section, we describe our experimental setup and show that ATOM can substantially improve OOD detection performance on both clean OOD data and adversarially perturbed OOD inputs. We also conducted extensive ablation analysis to explore different aspects of our algorithm.

5.4.1 Setup

In-distribution datasets. We use SVHN (Netzer et al., 2011), CIFAR-10 and CIFAR-100 (Krizhevsky et al., 2009) as in-distribution datasets.

Auxiliary OOD datasets. By default, we use 80 Million Tiny Images (TinyImages) (Torralba et al., 2008) as $\mathcal{D}_{\text{out}}^{\text{auxiliary}}$, which is a common setting in prior works. We also use ImageNet-RC, a variant of ImageNet (Chrabaszcz et al., 2017) as an alternative auxiliary OOD dataset.

Out-of-distribution datasets. For OOD test dataset, we follow common setup in literature and use six diverse datasets: `SVHN`, `Textures` (Cimpoi et al., 2014), `Places365` (Zhou et al., 2017), `LSUN (crop)`, `LSUN (resize)` (Yu et al., 2015), and `iSUN` (Xu et al., 2015).

Hyperparameters. The hyperparameter q is chosen on a separate validation set from TinyImages, which is different from test-time OOD data. Based on the validation, we set $q = 0.125$ for CIFAR-10 and $q = 0.5$ for CIFAR-100. For all experiments, we set $\lambda = 1$. For CIFAR-10 and CIFAR-100, we set $N = 400,000$, and $n = 100,000$.

Robust OOD evaluation tasks. We consider the following family of OOD inputs:

- **Natural OOD**: This is equivalent to the classic OOD evaluation with clean OOD input x, and $\Omega = \emptyset$.
- L_∞ **attacked OOD (white-box)**: We consider small L_∞-norm bounded perturbations on an OOD input x (Athalye et al., 2018; Madry et al., 2018), which induce the model to produce a high confidence score (or a low OOD score) for x. We denote the adversarial perturbations by $\Omega_{\infty,\epsilon}(x)$, where ϵ is the adversarial budget. We provide attack algorithms for all eight OOD detection methods in Appendix C.2.
- **Corruption attacked OOD (black-box)**: We consider a more realistic type of attack based on common corruptions (Hendrycks and Dietterich, 2019), which could appear naturally in the physical world. For each OOD image, we generate 75 corrupted images (15 corruption types $\times$ 5 severity levels), and then select the one with the lowest OOD score.
- **Compositionally attacked OOD (white-box)**: Lastly, we consider applying L_∞-norm bounded attack and corruption attack jointly to an OOD input x, as considered in Laidlaw and Feizi (2019a).

Evaluation metrics. We measure the following metrics: the false positive rate (FPR) at 5% false negative rate (FNR), and the area under the receiver operating characteristic curve (AUROC).

5.4.2 Results

ATOM vs. existing methods. We show in Table 5.1 that ATOM outperforms competitive OOD detection methods on both classic and adversarial OOD evaluation tasks. There are several salient observations. **First**, on classic OOD evaluation task (clean OOD data), ATOM achieves comparable or often even better performance than the current state-of-the-art methods. **Second**, on the existing adversarial OOD evaluation task, L_∞ OOD, ATOM outperforms current state-of-the-art method

$\mathcal{D}_{\text{in}}^{\text{test}}$	Method	FPR (5% FNR) ↓	AUROC ↑	FPR (5% FNR) ↓	AUROC ↑	FPR (5% FNR) ↓	AUROC ↑	FPR (5% FNR) ↓	AUROC ↑
		Natural OOD		Corruption OOD		L_∞ OOD		Comp. OOD	
SVHN	MSP (Hendrycks and Gimpel, 2017)	38.84	93.57	99.68	68.48	99.89	1.39	100.00	0.19
	ODIN (Liang et al., 2018)	31.45	93.52	97.11	63.21	99.86	0.61	100.00	0.05
	Mahalanobis (Lee et al., 2018b)	22.80	95.57	93.14	60.78	97.33	8.89	99.89	0.23
	SOFL (Mohseni et al., 2020)	**0.06**	**99.98**	**3.78**	**99.07**	75.31	46.78	99.81	2.75
	OE (Hendrycks et al., 2019b)	0.60	99.88	23.44	96.23	69.36	52.19	99.65	1.27
	ACET (Hein et al., 2019)	0.49	99.91	17.03	97.23	29.33	86.75	99.85	5.13
	CCU (Meinke and Hein, 2020)	0.50	99.90	24.17	96.11	52.17	62.24	99.42	1.60
	ROWL (Sehwag et al., 2019)	2.04	98.87	55.03	72.37	77.24	61.27	99.79	**50.00**
	ATOM (ours)	0.07	99.97	5.47	98.52	**7.02**	**98.00**	**96.33**	49.52
CIFAR-10	MSP (Hendrycks and Gimpel, 2017)	50.54	91.79	100.00	58.35	100.00	13.82	100.00	13.67
	ODIN (Liang et al., 2018)	21.65	94.66	99.37	51.44	99.99	0.18	100.00	0.01
	Mahalanobis (Lee et al., 2018b)	26.95	90.30	91.92	43.94	95.07	12.47	99.88	1.58
	SOFL (Mohseni et al., 2020)	2.78	99.04	62.07	88.65	99.98	1.01	100.00	0.76
	OE (Hendrycks et al., 2019b)	3.66	98.82	56.25	90.66	99.94	0.34	99.99	0.16
	ACET (Hein et al., 2019)	12.28	97.67	66.93	88.43	74.45	78.05	96.88	53.71
	CCU (Meinke and Hein, 2020)	3.39	98.92	56.76	89.38	99.91	0.35	99.97	0.21
	ROWL (Sehwag et al., 2019)	25.03	86.96	94.34	52.31	99.98	49.49	100.00	49.48
	ATOM (ours)	**1.69**	**99.20**	**25.26**	**95.29**	**20.55**	**88.94**	**38.89**	**86.71**
CIFAR-100	MSP (Hendrycks and Gimpel, 2017)	78.05	76.11	100.00	30.04	100.00	2.25	100.00	2.06
	ODIN (Liang et al., 2018)	56.77	83.62	100.00	36.95	100.00	0.14	100.00	0.00
	Mahalanobis (Lee et al., 2018b)	42.63	87.86	95.92	42.96	95.44	15.87	99.86	2.08
	SOFL (Mohseni et al., 2020)	43.36	91.21	99.93	45.23	100.00	0.35	100.00	0.27
	OE (Hendrycks et al., 2019b)	49.21	88.05	99.96	45.01	100.00	0.94	100.00	0.59
	ACET (Hein et al., 2019)	50.93	89.29	99.53	54.19	76.27	59.45	99.71	38.63
	CCU (Meinke and Hein, 2020)	43.04	90.95	99.90	48.34	100.00	0.75	100.00	0.48
	ROWL (Sehwag et al., 2019)	93.35	53.02	100.00	49.69	100.00	49.69	100.00	49.69
	ATOM (ours)	**32.30**	**93.06**	**93.15**	**71.96**	**38.72**	**88.03**	**93.44**	**69.15**

Table 5.1: Comparison with competitive OOD detection methods. We use DenseNet as network architecture for all methods. We evaluate on four types of OOD inputs: (1) natural OOD, (2) corruption attacked OOD, (3) L_∞ attacked OOD, and (4) compositionally attacked OOD inputs. The description of these OOD inputs can be found in Section 5.4.1. ↑ indicates larger value is better, and ↓ indicates lower value is better. All values are percentages and are averaged over six different OOD test datasets described in Section 5.4.1. **Bold** numbers are superior results.

ACET (Hein et al., 2019) by a large margin (e.g. on CIFAR-10, our method outperforms ACET by **53.9%** measured by FPR). **Third**, while ACET is somewhat brittle under the new Corruption OOD evaluation task, our method can generalize surprisingly well to the unknown corruption attacked OOD inputs, outperforming the best baseline by a large margin (e.g. on CIFAR-10, by up to **30.99%** measured by FPR). **Finally**, while almost every method fails under the hardest compositional OOD evaluation task, our method still achieves impressive results (e.g. on CIFAR-10, reduces the FPR by **57.99%**). The performance is noteworthy since our method is not trained explicitly on corrupted OOD inputs. Consistent performance improvement is observed on *alternative in-distribution datasets* (SVHN and CIFAR-100).

$\mathcal{D}_{\text{in}}^{\text{test}}$	Method	FPR (5%FNR) ↓	AUROC ↑	FPR (5% FNR) ↓	AUROC ↑	FPR (5%FNR) ↓	AUROC ↑	FPR (5%FNR) ↓	AUROC ↑
		Natural OOD		**Corruption OOD**		L_∞ **OOD**		**Comp. OOD**	
CIFAR-10	AT (no outlier mining)	2.65	99.11	42.28	91.94	44.31	68.64	65.17	72.62
	NTOM (no adversarial training)	1.87	**99.28**	30.58	94.67	99.90	1.22	99.99	0.45
	ATOM (ours)	**1.69**	99.20	**25.26**	**95.29**	**20.55**	**88.94**	**38.89**	**86.71**
CIFAR-100	AT (no outlier mining)	51.50	89.62	99.70	58.61	70.33	58.84	99.80	34.98
	NTOM (no adversarial training)	36.94	92.61	98.17	65.70	99.97	0.76	100.00	0.16
	ATOM (ours)	**32.30**	**93.06**	**93.15**	**71.96**	**38.72**	**88.03**	**93.44**	**69.15**

Table 5.2: **Ablation** on ATOM training objective. We use DenseNet as network architecture. ↑ indicates larger value is better, and ↓ indicates lower value is better. All values are percentages and are averaged over six different OOD test datasets described in Section 5.4.1.

Adversarial training alone is not able to achieve strong OOD robustness. We perform an ablation study that isolates the effect of outlier mining. In particular, we use the same training objective as in Equation (5.1), but with randomly sampled outliers. The results in Table 5.2 show AT (no outlier mining) is in general less robust. For example, under L_∞ OOD, AT displays 23.76% and 31.61% reduction in FPR on CIFAR-10 and CIFAR-100 respectively. This validates the importance of outlier mining for robust OOD detection, which provably improves the decision boundary as we will show in Section 5.5.

Effect of adversarial training. We perform an ablation study that isolates the effect of adversarial training. In particular, we consider the following objective without adversarial training:

$$\underset{\theta}{\text{minimize}} \quad \mathbb{E}_{(x,y)\sim\mathcal{D}_{\text{in}}^{\text{train}}}[\ell(x, y; \hat{F}_\theta)] + \lambda \cdot \mathbb{E}_{x\sim\mathcal{D}_{\text{out}}^{\text{train}}}[\ell(x, K+1; \hat{F}_\theta)], \tag{5.4}$$

which we name *Natural Training with informative Outlier Mining* (NTOM). In Table 5.2, we show that NTOM achieves comparable performance as ATOM on natural OOD and corruption OOD. However, NTOM is less robust under L_∞ OOD (with 79.35% reduction in FPR on CIFAR-10) and compositional OOD inputs. This underlies the importance of having both adversarial training and outlier mining (ATOM) for overall good performance, particularly for robust OOD evaluation tasks.

$\mathcal{D}_{\text{in}}^{\text{test}}$	Model	FPR (5% FNR) ↓	AUROC ↑	FPR (5% FNR) ↓	AUROC ↑	FPR (5% FNR) ↓	AUROC ↑	FPR (5% FNR) ↓	AUROC ↑
		Natural OOD		Corruption OOD		L_∞ OOD		Comp. OOD	
SVHN	ATOM (q=0.0)	0.07	99.97	5.47	98.52	7.02	98.00	96.33	49.52
	ATOM (q=0.125)	1.30	99.63	34.97	94.97	39.61	82.92	99.92	6.30
	ATOM (q=0.25)	1.36	99.60	41.98	94.30	52.39	71.34	99.97	1.35
	ATOM (q=0.5)	2.11	99.46	44.85	93.84	59.72	65.59	99.97	3.15
	ATOM (q=0.75)	2.91	99.26	51.33	93.07	66.20	57.16	99.96	2.04
CIFAR-10	ATOM (q=0.0)	2.24	99.20	40.46	92.86	36.80	73.11	66.15	73.93
	ATOM (q=0.125)	1.69	99.20	25.26	95.29	20.55	88.94	38.89	86.71
	ATOM (q=0.25)	2.34	99.12	22.71	95.29	24.93	94.83	41.58	91.56
	ATOM (q=0.5)	4.03	98.97	33.93	93.51	22.39	95.16	45.11	90.56
	ATOM (q=0.75)	5.35	98.77	41.02	92.78	21.87	93.37	43.64	91.98
CIFAR-100	ATOM (q=0.0)	44.38	91.92	99.76	60.12	68.32	65.75	99.80	49.85
	ATOM (q=0.125)	26.91	94.97	98.35	71.53	34.66	87.54	98.42	68.52
	ATOM (q=0.25)	32.43	93.93	97.71	72.61	40.37	82.68	97.87	65.19
	ATOM (q=0.5)	32.30	93.06	93.15	71.96	38.72	88.03	93.44	69.15
	ATOM (q=0.75)	38.56	91.20	97.59	58.53	62.66	78.70	97.97	54.89

Table 5.3: Ablation study on q. We use DenseNet as network architecture. ↑ indicates larger value is better, and ↓ indicates lower value is better. All values are percentages and are averaged over six natural OOD test datasets mentioned in Section 5.4.1. Note: the hyperparameter q is chosen on a separate validation set, which is different from test-time OOD data.

Effect of sampling parameter q**.** Table 5.3 shows the performance with different sampling parameter q. For all three datasets, training on auxiliary outliers with large OOD scores (*i.e.*, too easy examples with $q = 0.75$) worsens the performance, which suggests the necessity to include examples on which the OOD detector is uncertain. Interestingly, in the setting where the in-distribution data and auxiliary OOD data are disjoint (*e.g.*, SVHN/TinyImages), $q = 0$ is optimal, which suggests that the hardest outliers are mostly useful for training. However, in a more realistic setting, the auxiliary OOD data can almost always contain data similar to in-distribution data (*e.g.*, CIFAR/TinyImages). Even without removing near-duplicates exhaustively, ATOM can adaptively avoid training on those near-duplicates of in-distribution data (e.g. using $q = 0.125$ for CIFAR-10 and $q = 0.5$ for CIFAR-100).

Ablation on a different auxiliary dataset. To see the effect of the auxiliary dataset, we additionally experiment with ImageNet-RC as an alternative. From results in Table 5.4, we observe a consistent improvement of ATOM, and in many cases with

$\mathcal{D}_{\text{in}}^{\text{test}}$	Method	FPR (5% FNR) ↓	AUROC ↑	FPR (5% FNR) ↓	AUROC ↑	FPR (5% FNR) ↓	AUROC ↑	FPR (5% FNR) ↓	AUROC ↑
		Natural OOD		**Corruption OOD**		L_∞ **OOD**		**Comp. OOD**	
SVHN	MSP	38.84	93.57	99.68	68.48	99.89	1.39	100.00	0.19
	ODIN	31.45	93.52	97.11	63.21	99.86	0.61	100.00	0.05
	Mahalanobis	22.80	95.57	93.14	60.78	97.33	8.89	99.89	0.23
	SOFL	**0.02**	**99.99**	**5.93**	**98.57**	58.53	68.85	67.34	61.42
	OE	0.13	99.96	15.76	97.51	68.76	49.57	98.80	6.21
	ACET	0.31	99.94	29.02	95.65	2.37	99.51	30.58	95.20
	CCU	0.17	99.96	18.64	96.94	45.38	69.14	92.30	20.88
	ROWL	2.04	98.87	55.03	72.37	77.24	61.27	99.79	50.00
	ATOM (ours)	**0.02**	**99.99**	7.03	98.38	**0.14**	**99.95**	**7.30**	**98.32**
CIFAR-10	MSP	50.54	91.79	100.00	58.35	100.00	13.82	100.00	13.67
	ODIN	21.65	94.66	99.37	51.44	99.99	0.18	100.00	0.01
	Mahalanobis	26.95	90.30	91.92	43.94	95.07	12.47	99.88	1.58
	SOFL	6.96	98.71	22.30	95.89	97.61	12.39	99.74	7.49
	OE	9.70	98.35	49.84	91.76	91.30	43.88	98.82	31.12
	ACET	10.72	98.01	53.85	90.19	17.10	96.01	55.21	89.78
	CCU	10.30	98.25	44.42	92.34	93.02	20.88	99.17	9.95
	ROWL	25.03	86.96	94.34	52.31	99.98	49.49	100.00	49.48
	ATOM (ours)	**4.08**	**99.14**	**16.17**	**96.94**	**7.46**	**98.50**	**18.35**	**96.60**
CIFAR-100	MSP	78.05	76.11	100.00	30.04	100.00	2.25	100.00	2.06
	ODIN	56.77	83.62	100.00	36.95	100.00	0.14	100.00	0.00
	Mahalanobis	42.63	87.86	95.92	42.96	95.44	15.87	99.86	2.08
	SOFL	20.95	96.06	73.33	83.31	93.41	12.90	99.98	3.36
	OE	18.52	95.27	86.83	66.95	96.27	18.79	99.97	4.88
	ACET	19.79	94.76	81.63	70.04	26.23	91.46	81.95	69.67
	CCU	19.44	95.05	84.11	69.09	84.89	35.85	99.61	15.67
	ROWL	93.35	53.02	100.00	49.69	100.00	49.69	100.00	49.69
	ATOM (ours)	**15.49**	**97.18**	**57.79**	**89.49**	**18.32**	**96.57**	**58.49**	**89.36**

Table 5.4: Comparison with competitive OOD detection methods. We use ImageNet-RC as the auxiliary OOD dataset for SOFL, OE, ACET, CCU, NTOM and ATOM. We use DenseNet as network architecture for all methods. We evaluate on four types of OOD inputs: (1) natural OOD, (2) corruption attacked OOD, (3) L_∞ attacked OOD, and (4) compositionally attacked OOD inputs. ↑ indicates larger value is better, and ↓ indicates lower value is better. All values are percentages and are averaged over six natural OOD test datasets described in Section 5.4.1. **Bold** numbers are superior results.

performance better than using TinyImages. For example, on CIFAR-100, the FPR under natural OOD inputs is reduced from 32.30% (w/ TinyImages) to 15.49% (w/ ImageNet-RC). Interestingly, in all three datasets, using $q = 0$ (hardest outliers) yields the optimal performance since there are substantially fewer near-duplicates between ImageNet-RC and in-distribution data. This ablation suggests that ATOM's success does not depend on a particular auxiliary dataset.

5.5 Theoretical Analysis

In this section, we provide theoretical insight on mining informative outliers for robust OOD detection. We proceed with a brief summary of our key results.

Results overview. At a high level, our analysis provides two important insights. **First**, we show that with informative auxiliary OOD data, *less* in-distribution data is needed to build a robust OOD detector. **Second**, we show using outlier mining achieves a robust OOD detector in a more *realistic* case when the auxiliary OOD data contains many outliers that are far from the decision boundary (and thus non-informative), and may contain some in-distribution data. The above two insights are important for building a robust OOD detector in practice, particularly because labeled in-distribution data is expensive to obtain while auxiliary outlier data is relatively cheap to collect. *By performing outlier mining, one can effectively reduce the sample complexity while achieving strong robustness.* We provide the main results and intuition here and refer readers to Appendix C.1 for the details and the proofs.

5.5.1 Setup

Data model. To establish formal guarantees, we use a Gaussian $\mathcal{N}(\mu, \sigma^2 I)$ to model the in-distribution P_X and the test OOD distribution can be any distribution largely supported outside a ball around μ. We consider robust OOD detection under adversarial perturbation with bounded ℓ_∞ norm, i.e., the perturbation $\|\delta\|_\infty \leqslant \epsilon$. Given $\mu \in \mathbb{R}^d, \sigma > 0, \gamma \in (0, \sqrt{d}), \epsilon_\tau > 0$, we consider the following data model:

- P_X (**in-distribution data**) is $\mathcal{N}(\mu, \sigma^2 I)$. The in-distribution data $\{x_i\}_{i=1}^n$ is drawn from P_X.
- Q_X (**out-of-distribution data**) can be any distribution from the family $\mathcal{Q} = \{Q_X : \Pr_{x\sim Q_X}[\|x - \mu\|_2 \leqslant \tau] \leqslant \epsilon_\tau\}$, where $\tau = \sigma\sqrt{d} + \sigma\gamma + \epsilon\sqrt{d}$.
- **Hypothesis class of OOD detector**: $\mathcal{G} = \{G_{u,r}(x) : G_{u,r}(x) = 2 \cdot \mathbb{I}[\|x - u\|_2 \leqslant r] - 1, u \in \mathbb{R}^d, r \in \mathbb{R}_+\}$.

Here, γ is a parameter indicating the margin between the in-distribution and OOD data, and ϵ_τ is a small number bounding the probability mass the OOD distribution can have close to the in-distribution.

Metrics. For a detector G, we are interested in the False Negative Rate $\mathrm{FNR}(G)$ and the worst False Positive Rate $\sup_{Q_X \in \mathcal{Q}} \mathrm{FPR}(G; Q_X, \Omega_{\infty,\epsilon}(x))$ over all the test OOD distributions $\mathcal{Q}$ under ℓ_∞ perturbations of magnitude ϵ. For simplicity, we denote them as $\mathrm{FNR}(G)$ and $\mathrm{FPR}(G; \mathcal{Q})$.

While the Gaussian data model may be simpler than the practical data, its simplicity is desirable for our purpose of demonstrating our insights. Finally, the analysis can be generalized to mixtures of Gaussians which better models real-world data.

5.5.2 Learning with Informative Auxiliary Data

We show that informative auxiliary outliers can reduce the sample complexity for in-distribution data. Note that learning a robust detector requires to estimate μ to distance $\gamma\sigma$, which needs $\tilde{\Theta}(d/\gamma^2)$ in-distribution data, for example, one can compute a robust detector by:

$$u = \bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i, \quad r = (1 + \gamma/4\sqrt{d})\hat{\sigma}, \tag{5.5}$$

where $\hat{\sigma}^2 = \frac{1}{n}\sum_{i=1}^{n} \|x_i - \bar{x}\|_2^2$. Then we show that with informative auxiliary data, we need much less in-distribution data for learning. We model the auxiliary data U_X as a distribution over the sphere $\{x : \|x - \mu\|_2^2 = \sigma_o^2 d\}$ for $\sigma_o > \sigma$, and assume its density is at least η times that of the uniform distribution on the sphere for some constant $\eta > 0$, i.e., it's surrounding the boundary of P_X. Given $\{x_i\}_{i=1}^{n}$ from P_X and $\{\tilde{x}_i\}_{i=1}^{n'}$ from U_X, a natural idea is to compute $\bar{x}$ and r as above as an intermediate solution, and refine it to have small errors on the auxiliary data under perturbation,

i.e., find u by minimizing a natural "margin loss":

$$u = \underset{p:\|p-\bar{x}\|_2 \leqslant s}{\arg\min} \frac{1}{n'} \sum_{i=1}^{n'} \max_{\|\delta\|_\infty \leqslant \epsilon} \mathbb{I}\left[\|\tilde{x}_i + \delta - p\|_2 < t\right] \tag{5.6}$$

where s, t are hyper-parameters to be chosen. We show that with $\tilde{O}(d/\gamma^4)$ in-distribution data and sufficient auxiliary data can give a robust detector. See proof in Appendix C.1.2.

5.5.3 Learning with Informative Outlier Mining

In this subsection, we consider a more realistic data distribution where the auxiliary data can contain non-informative outliers (far away from the boundary), and in some cases mixed with in-distribution data. The non-informative outliers may not provide useful information to distinguish a good OOD detector statistically, which motivates the need for outlier mining.

Uninformative outliers can lead to bad detectors. To formalize, we model the non-informative ("easy" outlier) data as $Q_q = \mathcal{N}(0, \sigma_q^2 I)$, where σ_q is large to ensure they are obvious outliers. The auxiliary data distribution U_{mix} is then a mixture of U_X, Q_q and P_X, where Q_q has a large weight. Formally, $U_{mix} = \nu U_X + (1-2\nu)Q_q + \nu P_X$ for a small $\nu \in (0, 1)$. Then we see that the previous learning rule cannot work: those robust detectors (with u of distance $O(\sigma\gamma)$ to μ) and those bad ones (with u far away from μ) cannot be distinguished. There is only a small fraction of auxiliary data from U_X for distinguishing the good and bad detectors, while the majority (those from Q_q) do not differentiate them and some (those from P_X) can even penalize the good ones and favor the bad ones.

Informative outlier mining improves the detector with reduced sample complexity. The above failure case suggests that a more sophisticated method is needed. Below we show that outlier mining can help to identify informative data and improve the learning performance. It can remove most data outside U_X, and keep the

data from U_X, and the previous method can work after outlier mining. We first use in-distribution data to get an intermediate solution $\bar{x}$ and r by equations (5.5). Then, we use a simple thresholding mechanism to only pick points close to the decision boundary of the intermediate solution, which removes *non-informative outliers*. Specifically, we only select outliers with mild "confidence scores" w.r.t. the intermediate solution, i.e., the distances to $\bar{x}$ fall in some interval $[a, b]$:

$$S := \{i : \|\tilde{x}_i - \bar{x}\|_2 \in [a, b], 1 \leqslant i \leqslant n'\} \tag{5.7}$$

The final solution u_{om} is obtained by solving (5.6) on only S instead of all auxiliary data. We can prove:

Proposition 5. (***Error bound with outlier mining.***) *Suppose* $\sigma^2\gamma^2 \geqslant C\epsilon\sigma_o d$ *and* $\sigma\sqrt{d} + C\sigma\gamma^2 < \sigma_o\sqrt{d} < C\sigma\sqrt{d}$ *for a sufficiently large constant* C*, and* $\sigma_q\sqrt{d} > 2(\sigma_o\sqrt{d} + \|\mu\|_2)$*. For some absolute constant* c *and any* $\alpha \in (0, 1)$*, if the number of in-distribution data* $n \geqslant \frac{Cd}{\gamma^4}\log\frac{1}{\alpha}$ *and the number of auxiliary data* $n' \geqslant \frac{\exp(C\gamma^4)}{\nu^2\eta^2}\log\frac{d\sigma}{\alpha}$*, then there exist parameter values* s, t, a, b *such that with probability* $\geqslant 1 - \alpha$*, the detector* $G_{u_{om},r}$ *computed above satisfies:*

$$\mathrm{FNR}(G_{u_{om},r}) \leqslant \exp(-c\gamma^2), \quad \mathrm{FPR}(G_{u_{om},r}; Q) \leqslant \epsilon_\tau.$$

This means that even in the presence of a large amount of uninformative or even harmful auxiliary data, we can successfully learn a good detector. Furthermore, this can reduce the sample size n by a factor of γ^2. For example, when $\gamma = \Theta(d^{1/8})$, we only need $n = \tilde{\Theta}(\sqrt{d})$, while in the case without auxiliary data, we need $n = \tilde{\Theta}(d^{3/4})$.

Remark. We note that when U_X is as ideal as the uniform distribution over the sphere (i.e., $\eta = 1$), then we can let u be the average of points in S after mining, which will require $n' = \tilde{\Theta}(d/(\nu^2\gamma^2))$ auxiliary data, much less than that for more general η. We also note that our analysis and the result also hold for many other

auxiliary data distributions $\mathcal{U}_{\text{mix}}$, and the particular $\mathcal{U}_{\text{mix}}$ used here is for the ease of explanation; see Appendix C.1 for more discussions.

5.6 Related Work

OOD detection. Hendrycks and Gimpel (2017) introduced a baseline for OOD detection using the maximum softmax probability from a pre-trained network. Subsequent works improve the OOD uncertainty estimation by using deep ensembles (Lakshminarayanan et al., 2017), the calibrated softmax score (Liang et al., 2018), the Mahalanobis distance-based confidence score (Lee et al., 2018b), as well as the energy score (Liu et al., 2020). Some methods regularize the model with auxiliary anomalous data that were either realistic (Hendrycks et al., 2019b; Mohseni et al., 2020; Papadopoulos et al., 2021) or artificially generated by GANs (Lee et al., 2018a). Several other works (Bevandić et al., 2018; Malinin and Gales, 2018; Subramanya et al., 2017) also explored regularizing the model to produce lower confidence for anomalous examples. Recent works have also studied the computational efficiency aspect of OOD detection (Lin et al., 2021) and large-scale OOD detection on ImageNet (Huang and Li, 2021).

Robustness of OOD detection. Worst-case aspects of OOD detection have been studied in Hein et al. (2019); Sehwag et al. (2019). However, these papers are primarily concerned with L_∞ norm bounded adversarial attacks, while our evaluation also includes common image corruption attacks. Besides, Hein et al.; Meinke and Hein only evaluate adversarial robustness of OOD detection on random noise images, while we also evaluate it on natural OOD images. Meinke and Hein have shown the first provable guarantees for worst-case OOD detection on some balls around uniform noise, and Bitterwolf et al. studied the provable guarantees for worst-case OOD detection not only for noise but also for images from related but different image classification tasks. This chapter proposes ATOM which achieves state-of-the-art performance on a broader family of clean and perturbed OOD inputs. The key difference compared to prior work is introducing the informative

outlier mining technique, which can significantly improve the generalization and robustness of OOD detection.

Adversarial robustness. Adversarial examples (Biggio et al., 2013a; Goodfellow et al., 2015; Papernot et al., 2016; Szegedy et al., 2014) have received considerable attention in recent years. Many defense methods have been proposed to mitigate this problem. One of the most effective methods is adversarial training (Madry et al., 2018), which uses robust optimization techniques to render deep learning models resistant to adversarial attacks. Carmon et al.; Najafi et al.; Uesato et al.; Zhai et al. showed that unlabeled data could improve adversarial robustness for classification.

Hard example mining. Hard example mining was introduced in Sung (1995) for training face detection models, where they gradually grew the set of background examples by selecting those examples for which the detector triggered a false alarm. The idea has been used extensively for object detection literature (Felzenszwalb et al., 2009; Gidaris and Komodakis, 2015; Shrivastava et al., 2016). It also has been used extensively in deep metric learning (Cui et al., 2016; Harwood et al., 2017; Simo-Serra et al., 2015; Suh et al., 2019; Wang and Gupta, 2015) and deep embedding learning (Duan et al., 2019; Smirnov et al., 2018; Wu et al., 2017; Yuan et al., 2017). Although hard example mining has been used in various learning domains, to the best of our knowledge, we are the first to explore it to improve the robustness of out-of-distribution detection.

5.7 Conclusion

In this chapter, we proposed Adversarial Training with informative Outlier Mining (ATOM), a method that enhanced the robustness of the OOD detector. We showed the merit of adaptively selecting the OOD training examples which the OOD detector was uncertain about. Extensive experiments showed ATOM could significantly improve the decision boundary of the OOD detector, achieving state-of-the-art

performance under a broad family of *clean and perturbed* OOD evaluation tasks. We also provided a theoretical analysis that justified the benefits of outlier mining. Further, our unified evaluation framework allowed future research to examine the robustness of the OOD detector. We hoped our research could raise more attention to a broader view of robustness in out-of-distribution detection.

6 DETECTING ERRORS AND ESTIMATING ACCURACY ON UNLABELED DATA WITH SELF-TRAINING ENSEMBLES

Contribution statement. This chapter is joint work with Frederick Liu, Besim Avci, Xi Wu, Yingyu Liang, and Somesh Jha. The author Jiefeng Chen proposed the method, contributed to part of the theoretical analysis, and completed all the experiments. The paper version of this chapter appeared in NeurIPS 2021 (Chen et al., 2021b).

6.1 Introduction

Data distribution in the real world may be wildly different from the training dataset for various reasons such as covariate shift due to domain divergence, corruption of images due to weather conditions, or out-of-distribution test inputs. When facing these issues, a deployed deep learning model can have unexpected performance drop on the test data. This performance degradation can be mitigated by test data annotation, which may be costly. Hence, estimating the accuracy of a pre-trained model on the unlabeled test data provides an alternative to avoid the cost when it is not necessary. Furthermore, it is beneficial to estimate the correctness of the predictions on individual points. This leads to an even more challenging task of error detection, which aims to identify points in the unlabeled test set that are mis-classified by the pre-trained model. Such a finer-grained estimation can facilitate a further improvement of the pre-trained model (e.g., manually label those mis-classified data points and retrain the model on them).

While there have been previous attempts to address accuracy estimation or the broader problem of error detection, their successes usually rely on some conditions or assumptions that may not hold in practice. For example, a natural approach is to use confidence based metrics to measure the performance of the pre-trained model, e.g., as in Elsahar and Gallé (2019). If the model is well-calibrated on the test data, then the average confidence approximates its accuracy. However, it has

been observed that many machine learning systems, in particular modern neural networks, are poorly calibrated, especially on test data with distribution shift (Guo et al., 2017; Ovadia et al., 2019). Another method is to learn a regression function that takes statistics about the model and the test data as input, and predicts the performance on the test data (Elsahar and Gallé, 2019; Schelter et al., 2020). This requires training on labeled data from various data distributions, which is very expensive or even impractical. Furthermore, the performance predictor trained on the labeled data may not generalize to unknown data distributions. Recent work by Chuang et al. (2020) proposes to learn a "check" model using domain-invariant representation and use it as a proxy for the unknown true test labels to estimate the performance via error detection. It relies on the success of the domain-invariant representation methods to obtain a highly accurate check model on the test data. Hence, the check model performance suffers when domain-invariant representation is not accurate in circumstances such as test data having outlier feature vectors or different class probabilities than the training data.

In this chapter, we propose a principled and practically effective framework for the challenging tasks of accuracy estimation and error detection (Section 6.4). The framework makes a novel use of the self-training technique on ensembles for these tasks. It first learns an ensemble of models to identify some mis-classified points. Then it assigns pseudo-labels to these points and uses self-training with these pseudo-labeled data to identify more mis-classified points. We also provide provable guarantees for the framework (Section 6.5). Our analysis shows that it provably outputs an accurate estimate of the accuracy as well as the mis-classified points, under mild practical conditions: the ensemble make small errors on the test inputs correctly classified by the model f, mostly disagree with f on the current identified mis-classified inputs, and are correct or have diverse predictions on the remaining inputs. These conditions can be readily satisfied by deep learning model ensembles. Furthermore, they have no explicit assumptions on the test distribution and thus the framework can be instantiated by incorporating properly designed ensemble methods for different settings (Section 6.6). Experimental results on 59 tasks over five dataset categories including image classification and

sentiment classification datasets show that our method achieves state-of-the-art on both accuracy estimation and error detection (Section 6.7). The experiments also provide positive support for our analysis, verifying the conditions and implications.

6.2 Related Work

Confidence Estimation and Error Detection. Recently, estimating model confidence has been an important area of research because of the perceived relationship between model uncertainty and trusting its predictions (Lo Piano, 2020). However, modern neural networks have been observed to be poorly calibrated (e.g. they may make wrong predictions with very high confidence) (Guo et al., 2017), especially on distributions with dataset shift (Ovadia et al., 2019). Many approaches have been proposed to address this issue, such as *Temperature Scaling* (Guo et al., 2017), *Monte-Carlo Dropout* (Gal and Ghahramani, 2016) and *Deep Ensemble* (Lakshminarayanan et al., 2017), but the challenge still remains. A similar challenge appears in error detection, where the goal is to identify erroneous predictions given a test set. Combining these two problems, some early work uses confidence estimates to detect incorrect predictions. For example, Hendrycks and Gimpel proposed to use maximum softmax probability to detect misclassified examples. Corbière et al. proposed *True Class Probability* for failure prediction. Jiang et al. proposed to use *Trust Score* to estimate the confidence in model predictions. These methods require a robust estimate of confidence, and an empirically chosen threshold that is mostly problem- and model-dependent, to identify error data points. Recently, Chuang et al. proposed to use a check model to predict mis-classification.

Unsupervised Accuracy Estimation. The problem of unsupervised accuracy (or risk) estimation has received relatively scant attention from the research community. Donmez et al. offered a solution with certain assumptions on the marginal output distribution $p(y)$. Platanios et al. proposed to estimate the accuracies of the approximations to some target Boolean functions based on the agreement rates method. A follow-up work by Platanios et al. (2017) also considered the "multiple

approximations" problem setting where the target classes may be tied together through logical constraints, and proposed an efficient method to estimate the accuracy of classifiers using only unlabeled data. Jaffe et al. proposed a spectral-based approach to estimate the accuracies of multiple classifiers, mainly in the binary case and with some assumptions on the classifiers. Steinhardt and Liang proposed a method to estimate the model's error on distributions different from the training distribution by assuming a conditional independence structure of the data. These problem settings are different from ours and the constraints may not be satisfied by the data and model we consider. Recently, Elsahar and Gallé studied three families of methods ($\mathcal{H}$-divergence, reverse classification accuracy and confidence measures) and demonstrated how they could be used to predict the performance drop. Schelter et al. also proposed to learn a performance predictor on the statistics of model outputs with an assumption that they know the typical cases of dataset shift in advance. Similarly, Deng and Zheng proposed to train regression models on the feature statistics of the datasets sampled from a meta-dataset to predict model performance and Deng et al. proposed to utilize linear regression to estimate classifier performance from the accuracy of rotation prediction. Guillory et al. proposed to use difference of confidences to estimate classifier performance. Chen et al. proposed MANDOLINE that utilizes user-specified slicing functions to improve the importance of weighting to make the accuracy estimation more accurate. With the most similar setup to ours, Chuang et al. used a set of domain-invariant predictors as a proxy for the unknown target labels to estimate a given model's performance under distribution shift. In a concurrent work, Jiang et al. empirically showed that the test accuracy of deep networks can be estimated by measuring disagreement rate between a pair of models independently trained via Stochastic Gradient Descent (SGD) and theoretically related this phenomenon to the well-calibrated nature of ensembles of SGD-trained models.

6.3 Problem Statement

Consider the classification problem with sample space $\mathcal{X}$ and label space $\mathcal{Y}$. Let $\mathcal{D}^{tr}$ be a set of labeled data from the training distribution $P_{X,Y}$. Let $U = \{(x, y_x)\}$ be a set of labeled data from the test distribution $Q_{X,Y}$, where y_x is the label for x. Let $U_X = \{x : (x, y_x) \in U\}$ be the set of input feature vectors in U. Define the accuracy of a model f on U as: $\text{acc}(f, U) := \frac{1}{|U|} \sum_{(x,y_x)\in U} \mathbb{I}[f(x) = y_x]$, where $|U|$ is the cardinality of the set U. When clear from the context, we will omit U and simply write $\text{acc}(f)$. Given a model $f : \mathcal{X} \to \mathcal{Y}$ trained on $\mathcal{D}^{tr}$, together with $\mathcal{D}^{tr}$ and U_X, the goal of *unsupervised accuracy estimation* is to get an estimate $\hat{\text{acc}}$ so that the absolute estimation error $|\hat{\text{acc}} - \text{acc}(f)|$ is small. We assume access to the training data $\mathcal{D}^{tr}$ to help in estimating $\text{acc}(f)$.

Though $\text{acc}(f)$ is usually sufficient to provide an estimate of whether the model could perform well on U, it would be beneficial if the algorithm could also provide an estimate of the correctness of individual points so that we can know where to improve the model f. Thus, we also consider a more challenging task of identifying the mis-classified points in U: $W_X := \{x : (x, y_x) \in U, f(x) \neq y_x\}$. Given f, $\mathcal{D}^{tr}$ and U_X, the goal of *error detection* is to identify a subset $R_X \subseteq U_X$, such that $|W_X \triangle R_X|$ is small, where $W_X \triangle R_X := (W_X \setminus R_X) \cup (R_X \setminus W_X)$.

6.4 Algorithmic Framework via Self-Training Ensembles

Intuition. We consider the approach of learning a "check model" h and using the disagreement between h and the pre-trained model f for our tasks. The fundamental idea is to identify a point x as mis-classified if h disagrees with f on x. To derive our method, we note a simple fact: the disagreement approach succeeds if (1) h agrees with f on points where f is correct and (2) h disagrees with f on points where f is incorrect. Our first key observation is that usually (1) can be satisfied approximately in practice. Intuitively, this is because h and f use the same training

data, and h can be trained to be correct on the subset of the instance space where f is correct. However, (2) may not be easily satisfied (e.g., h can make similar mistakes as f), which leads to an overestimation of the accuracy. We thus focus on improving the disagreement on mis-classified points.

To disagree with f on a mis-classified test input x, we have two ways: (1) make the check model h correct on x; (2) diverse ensembles. The first way may be achievable when the training data contains information for prediction on x. A prototypical example is when the test inputs are corruption of clean data from the training distribution (e.g., the training data are images in sunny days while the test inputs are ones in rainy days), and techniques like unsupervised domain adaptation can be used to improve the prediction on such test inputs. However, correct predictions on x may not be feasible in many interesting scenarios due to insufficient information (e.g., the test image in the open world can contain an object that is never seen in the training data). Fortunately, it has been shown that for such inputs, one can obtain an ensemble of models with diverse predictions (e.g.,Lakshminarayanan et al. (2017)). This then gives the second way to achieve disagreement: using diverse ensembles. Therefore, our method will learn an *ensemble* of models (instead of one check model) and identify a point x as mis-classified if the ensemble disagree with f on x (i.e., a large fraction of the models in the ensemble disagree with f on their predictions on x). A mis-classified test input, x, will be successfully identified, if a majority of the ensemble models predict correctly, or they have large diversity on x.

However, the ensemble may only be able to identify a subset of the mis-classified points. Therefore, we propose to iteratively identify more and more mis-classified points by *self-training*. For each mis-classified data point x identified by the ensemble, we assign it a pseudo-label that is different from $f(x)$ (e.g. use the majority vote of the ensemble or a random label as the pseudo-label). Then we can train (with regularization) a new ensemble to encourage their disagreement with f on the pseudo-labeled data R (e.g., use a supervised loss on R with a small weight as the regularization). Note that the pseudo-labels may not all be correct, and we do not need the new ensemble to exactly fit the pseudo-labels. We only need

Framework 5 Error Detection and Unsupervised Accuracy Estimation via Self-Training Ensembles

Require: A training dataset $\mathcal{D}^{\text{tr}}$, an unlabeled test dataset U_X, a pre-trained model f, an ensemble learning method $\mathcal{T}$, and a hyper-parameter τ

1: Initialize $R = \emptyset$
2: **for** $t = 1, 2, \cdots, T$ **do**
3: Use $\mathcal{T}$ to generate an ensemble $\mathcal{T}(\mathcal{D}^{\text{tr}}, U_X, R)$.
4: Identify those points on which the ensemble and f disagree as mis-classified points:

$$R_X := \{x \in U_X : \Pr_{h \sim \mathcal{T}}\{h(x) = f(x)\} < \tau\}.$$

5: For each $x \in R_X$, assign a pseudo-label $\tilde{y}_x \neq f(x)$ (e.g., using the majority vote of the ensemble or a random label). And set $R = \{(x, \tilde{y}_x) : x \in R_X\}$.
6: **end for**

Ensure: The estimated mis-classified points R_X, and the estimated accuracy $\frac{|U_X \setminus R_X|}{|U_X|}$.

the new ensemble to mostly disagree with f on R so that they still identify R as mis-classified points. Furthermore, self-training can help the ensemble identify more mis-classified points. When some pseudo-labels are correct, we observe that these additional data can help the new ensemble become more accurate and thus help with identifying more mis-classified points. We also observe that the new ensemble will be less diverse on the pseudo-labeled data and thus have diversity on the remaining test inputs.

Our framework. We assume access to an ensemble learning method $\mathcal{T}$ (specific instantiations will be given later) that takes as input a training set $\mathcal{D}^{\text{tr}}$, an unlabeled test set U_X, and a pseudo-labeled set R (and potentially some other parameters) and outputs an ensemble, which is a distribution over functions $h : \mathcal{X} \to \mathcal{Y}$.[1] We denote the generated ensemble as $\mathcal{T}(\mathcal{D}^{\text{tr}}, U_X, R)$ and write $h \sim \mathcal{T}(\mathcal{D}^{\text{tr}}, U_X, R)$ for sampling h

[1]This includes one single h as a special case. It also includes the case of h with probabilistic outputs, i.e., $h(x) = [h^1(x), \ldots, h^K(x)]$ for $\mathcal{Y} = \{1, 2, \ldots, K\}$, where $h^i(x)$ is the predicted probability of x from class i. One can think of such an h as an ensemble $\mathcal{T}$ where $\Pr_{g \sim \mathcal{T}}[g(x) = i] = h^i(x)$.

from the ensemble, or simply $h \sim \mathcal{T}$ when clear from context.

We are now ready to describe our method (Framework 5). Our framework begins with an empty set of pseudo-labeled data R, and executes in T iterations. In each iteration, it generates an ensemble $\mathcal{T}(\mathcal{D}^{tr}, U_X, R)$ and then uses the ensemble to construct a new pseudo-labeled dataset R for the next iteration: let R_X be those points $x \in U_X$ where the agreement rate between the ensemble and f is below a threshold, assign a pseudo-label $\tilde{y}_x$ different from $f(x)$, and let R be the set of these pseudo-labeled data. Finally, it outputs R_X as the mis-classified points and outputs the fraction of points outside R_X as the accuracy.

Some existing work also use models' disagreement to estimate the error on the unlabeled data in different ways. For example, Platanios et al. consider a set of pre-trained models and aim to estimate the error for each model; Chuang et al. learn a single check model and compare it with f. On the other hand, our novelty is using *ensembles* to identify a set of mis-classified points and further using *self-training* for learning the ensembles. Note that our self-training ensembles is different from standard self-training and ensemble: standard self-training+ensemble aims to get accurate predictions while ours is to increase disagreement on mis-classified points.

6.5 Theoretical Analysis

As described in the intuition above, our framework succeeds if in each iteration, the ensemble satisfy the following conditions: (**A**) correct on $U_X \setminus W_X$, (**B**) mostly disagree with f on R_X, and (**C**) either correct or diverse on $W_X \setminus R_X$. This section first introduces notions formalizing these conditions and then provides provable guarantees under these conditions. Here we focus on intuitions and provide the proofs and more discussion in Appendix D.1.

Notations. Recall that the ensemble method $\mathcal{T}$ outputs a distribution over models, and we write $h \sim \mathcal{T}$ for sampling a model h from this distribution. Let $\mathbb{E}_h$ denote the expectation over this output distribution, and $\mathbb{E}_x$ denote the average over $x \in U_X$ or $(x, y_x) \in U$.

Formalizing Condition (A). A key observation we make in practice is that usually on points where f is correct, the ensemble models are also correct. Intuitively, this is because the only labeled information for learning the ensemble is $\mathcal{D}^{\text{tr}}$, while $\mathcal{D}^{\text{tr}}$ is also used for learning f. For illustration, suppose f and the ensemble have zero training errors on $\mathcal{D}^{\text{tr}}$. When they are from hypothesis classes of bounded VC-dimensions and the training set is large enough compared to the VC-dimensions, standard error bounds show that the ensemble will have small errors on the subset of the feature space $\{x \in \mathcal{X} : f(x) = y_x\}$, i.e., it will have small errors on points where f is correct. To formalize this observation, we introduce the following notion.

Definition 6.1 (Error on Correct Points)**.** *Let ν denote the average probability of $h \sim \mathcal{T}$ making error on test inputs where the model f is correct:* $\nu := \Pr_{(x,y_x)\sim U, h\sim\mathcal{T}}[h(x) \neq y_x \mid f(x) = y_x]$.

Formalizing Condition (B). We assume the new ensemble trained with regularization on R will mostly disagree with f on R_X. We define the following notion:

Definition 6.2 (Agreement on Pseudo-Labeled Data)**.** *Let γ denote the average probability of agreement between $h \sim \mathcal{T}$ and f on R_X:* $\gamma := \Pr_{x\sim R_X, h\sim\mathcal{T}}\{h(x) = f(x)\}$.

Formalizing Condition (C). Let G_X denote the good points in $W_X \setminus R_X$ on which the ensemble will have correct predictions with high confidence, B_X the remaining bad points. Formally, we define: $G_X := \{x \in W_X \setminus R_X : \Pr_{h\sim\mathcal{T}}\{h(x) = y_x\} \geqslant 1 - \nu\}$ and $B_X := W_X \setminus (R_X \cup G_X)$.

(Here we choose the confidence level $1 - \nu$ for convenience, where ν is the error on correct points. Other sufficiently high confidence levels can also be used with slight change to the analysis.) We would like the ensemble to have large diversity on B_X.

Definition 6.3 (Diversity of Ensemble)**.** *Let σ^2 denote the average probability of disagreement between two ensemble models on B_X:* $\sigma^2 := \mathbb{E}_x[\sigma_x^2 | x \in B_X]$, *where* $\sigma_x^2 := \Pr_{h_1,h_2\sim\mathcal{T}}[h_1(x) \neq h_2(x)]$.

Provable Guarantees. Based on the above notions we obtain our guarantees:

Theorem 6.4. *Assume in each iteration of the framework,* $\tau = \sqrt{1-\eta}$ *for some* $\eta \in (0, 3B_\eta/4)$ *where* $B_\eta := \min\{\sigma^2, 1-\nu^2\}$. *Let* $\sigma_L^2 > 0$ *be a lower bound on the diversity* σ^2, $\tilde{\gamma} > 0$ *be an upper bound on* γ, *and* $\tilde{\nu}$ *be an upper bound on* ν *over all iterations. Then for any* $\delta \in (0, \sigma_L^2/4)$, *after at most* $\lceil 1/\delta \rceil$ *iterations, we can get* $\frac{|U_X \setminus R_X|}{|U_X|}$ *approximates the accuracy* $\mathrm{acc}(f)$ *and* R_X *approximates the mis-classified points* W_X *as follows:*

$$\left|\mathrm{acc}(f) - \frac{|U_X \setminus R_X|}{|U_X|}\right| \leqslant \max\{\frac{\tilde{\nu}}{1-\tau}(1-e_f), \epsilon e_f\}, \text{ where } \epsilon := \frac{\frac{\tilde{\gamma}}{\tau}\left(1 + \frac{\tilde{\nu}}{1-\tau}\frac{1-e_f}{e_f}\right)}{\frac{\sigma_L^2}{4} - \delta + \frac{\tilde{\gamma}}{\tau}}, \tag{6.1}$$

$$|W_X \triangle R_X| \leqslant \frac{\tilde{\nu}}{1-\tau}|U_X \setminus W_X| + \epsilon|W_X|. \tag{6.2}$$

Proof Sketch. Let's first consider one iteration, assuming small ν, γ and large σ^2. Intuitively, on the correct points $U_X \setminus W_X$, the ensemble have a small error ν and thus disagree with f on only a few such points. On the old R_X, the ensemble mostly disagree with $f(x)$; similarly on G_X. On the remaining points B_X, we show that if the ensemble have large diversity, then their predictions must have large variances on a significant subset of points and thus have large disagreement with f, facilitating the detection. Overall, our framework can construct a new pseudo-labeled set R'_X that contains mostly mis-classified points and is also larger than the old R_X (Lemma D.1 in Appendix D.1). Therefore, each iteration can make some progress by identifying more mis-classified points than before, and enough iterations achieve the guarantees. □

The theorem provides guarantees for general values of ν, γ and σ^2. To get some intuition, note that typically $e_f < \tau$ and suppose we set $\tau = 3/4$ and $\delta = \tilde{\gamma}/\tau$, then the accuracy is estimated up to error $\max\{\tilde{\nu}, \frac{16\tilde{\gamma}}{3\sigma_L^2}(e_f + \tilde{\nu})\}$. When $\tilde{\nu}, \tilde{\gamma}$ are small and σ_L^2 is large, the error is small. Therefore, under mild conditions, our framework can give a provable estimation of the accuracy and the mis-classified points up to small errors.

Algorithm 1 Error Detection and Unsupervised Accuracy Estimation via Self-Training Ensembles

Require: $\mathcal{D}^{\text{tr}}$, U_X, f, ensemble method $\mathcal{T}$, parameters T, N

1: Initialize $R = \emptyset$
2: **for** $t = 1, 2, \cdots, T$ **do**
3: Set $\{h_i\}_{i=1}^N = \mathcal{T}(\mathcal{D}^{\text{tr}}, U_X, R, N, \text{other parameters})$
4: Let $\tilde{y}_x$ be the majority vote of $\{h_i\}_{i=1}^N$: $\tilde{y}_x := \arg\max_{j \in \mathcal{Y}} \frac{1}{N} \sum_{i=1}^N \mathbb{I}[h_i(x) = j]$.
5: Set $R = \{(x, \tilde{y}_x) : x \in U_X, \tilde{y}_x \neq f(x)\}$.
6: **end for**

Ensure: Estimated mis-classified points $R_X = \{x : (x, y) \in R\}$, estimated accuracy $\frac{|U_X \setminus R_X|}{|U_X|}$.

The theorem formalizes that the framework can succeed under mild conditions **(A)(B)(C)** on the ensembles, without explicit conditions on the data distributions and the pre-trained model (note that the conditions on them are implicitly captured in the mild conditions on the ensemble). The framework is thus flexible, and different ensemble methods satisfying these conditions can be incorporated to get concrete instantiations applicable to various settings. Indeed, the crux of our framework is then to design ensemble methods meeting the conditions. This turns out to be not difficult, in particular for deep learning pre-trained models and deep learning ensemble models. Even an ensemble of deep networks simply trained from random initialization works well. Two concrete instantiations are presented in Section 6.6 below and evaluated in Section 6.7.

6.6 Instantiations of the Framework

Based on our framework, we propose Algorithm 1 for accuracy estimation and error detection. It executes in T iterations. In each iteration, it trains an ensemble of models $\{h_i\}_{i=1}^N$ and then uses the ensemble to construct the pseudo-labeled data. Finally, it outputs R_X as the set of mis-classified points and outputs the fraction of points outside R_X as the accuracy. The algorithm uses an intuitive heuristic to implement the threshold on the agreement rate: it sets R_X as the points $x \in U_X$

with $f(x)$ different from the majority vote of the ensemble models. Empirically, we observe that this leads to similar R_X and thus similar results as explicit thresholding, but is much more convenient.

While different ensemble methods $\mathcal{T}$ can be used in Algorithm 1, here we propose two concrete instantiations $\mathcal{T}_{RI}$ and $\mathcal{T}_{RM}$ based on the success conditions of our framework.

Ensemble Method $\mathcal{T}_{RI}$. Algorithm 2 describes a natural method that trains the models from different random initialization. It first trains h_i' on $\mathcal{D}^{tr}$ (e.g., using the same training algorithm as for f), to ensure that ensemble has small error on points where f is correct. It then fine-tunes h_i' on $\mathcal{D}^{tr}$ and R for one epoch to get h_i that mostly disagrees with f on R_X. Finally, deep models trained from different random initialization have been shown to be diverse on outlier data points (Lakshminarayanan et al., 2017; Fort et al., 2019). In summary, the ensemble constructed can satisfy our three conditions.

Ensemble Method $\mathcal{T}_{RM}$. Algorithm 3 describes another method designed with the representation matching technique for domain adaptation, which can potentially improve the accuracy of the ensemble on some test inputs related to the training data and thus satisfy our success condition (**C**) better. It requires the model architecture to be $c(\phi(x))$, i.e., a composition of a prediction function c and a representation function ϕ. Beginning from a pre-trained model h_0, it fine-tunes h_0 for N epochs by minimizing the loss on $\mathcal{D}^{tr}$ and R *plus a representation matching loss* $\alpha \cdot d(p^{\phi}_{\mathcal{D}^{tr}}, p^{\phi}_{U_X})$, and outputs the N checkpoint models at the end of each training epoch as the ensemble. A key component of $\mathcal{T}_{RM}$ is representation matching, which aims to learn a function ϕ that minimizes $d(p^{\phi}_{\mathcal{D}^{tr}}, p^{\phi}_{U_X})$. It has been shown that representation matching can improve the accuracy on the test data from the target domain. Also, we have observed that the checkpoint models can have diversity on the mis-classified data points empirically. Thus, the ensemble constructed can satisfy our conditions. For our experiments, we use the representation matching loss from the classic DANN (Ajakan et al., 2014).

Algorithm 2 $\mathcal{T}_{\text{RI}}$: Ensemble via Random Initialization

Require: $\mathcal{D}^{\text{tr}}$, U_X, R, N, parameter γ

1: Pre-train $\{h'_i\}_{i=1}^N$ on $\mathcal{D}^{\text{tr}}$ from different random initialization

2: **for** $i = 1, 2, \ldots, N$ **do**

3: Learn h_i by fine-tuning h'_i for one epoch by:

$$\underset{h}{\text{minimize}} \quad \mathbb{E}_{(x,y)\in\mathcal{D}^{\text{tr}}}[\ell(h(x), y)] + \gamma \cdot \mathbb{E}_{(x,y)\in R}[\ell(h(x), y)] \tag{6.3}$$

4: **end for**

Ensure: The ensemble of models $\{h_i\}_{i=1}^N$

Algorithm 3 $\mathcal{T}_{\text{RM}}$: Ensemble via Representation Matching

Require: $\mathcal{D}^{\text{tr}}$, U_X, R, N, initial pre-trained model h_0, parameters α, γ

// $h_0(x) = c(\phi(x))$ *is a composition of a prediction function* c *and a representation function* ϕ

1: Fine-tune h_0 for N epochs using the objective:

$$\underset{h}{\text{minimize}} \quad \mathbb{E}_{(x,y)\in\mathcal{D}^{\text{tr}}}[\ell(h(x), y)] + \gamma \cdot \mathbb{E}_{(x,y)\in R}[\ell(h(x), y)] + \alpha \cdot d(p^{\phi}_{\mathcal{D}^{\text{tr}}}, p^{\phi}_{U_X}) \tag{6.4}$$

where $d(p^{\phi}_{\mathcal{D}^{\text{tr}}}, p^{\phi}_{U_X})$ is the distance between the distribution of $\phi(x)$ on $\mathcal{D}^{\text{tr}}$ and that on U_X.

2: Use the N checkpoint models at the end of each training epoch as the model ensemble $\{h_i\}_{i=1}^N$.

Ensure: The ensemble of models $\{h_i\}_{i=1}^N$

6.7 Experiments

We perform experiments for unsupervised accuracy estimation and error detection tasks on 59 pairs of training-test datasets from five dataset categories, including image classification and sentiment classification datasets. In summary, our findings are: (**1**) Our method achieves state-of-the-art results on both accuracy estimation and error detection tasks than the existing methods. (**2**) Both ensemble and self-training techniques have positive effects on the tasks and it is easy to pick suitable hyper-parameters for our algorithms. (**3**) Empirical results show that the conditions

made in our analysis hold approximately.

6.7.1 Setup

We briefly describe the experiment setup here.

Dataset. The task needs a pair of training-test datasets $\mathcal{D}^{\text{tr}}$ and U_X. We use five dataset categories, each containing multiple training-test dataset pairs. Specifically, we use the following dataset categories: Digits (including MNIST (LeCun, 1998), MNIST-M (Ajakan et al., 2014), SVHN (Netzer et al., 2011), USPS (Hull, 1994)), Office-31 (Saenko et al., 2010), CIFAR10-C (Krizhevsky et al., 2009), iWildCam (Beery et al., 2020) and Amazon Review (Blitzer et al., 2007). Digits has 12 dataset pairs, Office-31 has 6, CIFAR10-C has 19, iWildCam has 10 and Amazon Review has 12.

Evaluation Metrics. We use absolute estimation error for accuracy estimation and use F1 score for error detection.

Our Models. On each dataset category, we design a neural network architecture for the DANN training algorithm (Ajakan et al., 2014), which is named DANN-arch. It contains an encoder, a predictor branch and a discriminator branch. For $\mathcal{T}_{\text{RM}}$, we use the DANN-arch for the ensemble $\{h_i\}$ and pre-train the initial model h_0 on $\mathcal{D}^{\text{tr}}$ and U_X using DANN algorithm. For $\{h_i\}$ in $\mathcal{T}_{\text{RI}}$, we also use the DANN-arch. The model f is pre-trained on D and we mainly consider two kinds of architectures for it: one is the DANN-arch (i.e., the model f shares the same architecture as the check model h); the other is a typical deep neural network (DNN).

Hyper-parameters. The hyper-parameters T, N and γ can be easily selected since a broad range of values can lead to good results. Based on our observation, larger T and N can lead to better results. In our experiments, we set $T = 5$ and $N = 5$ by considering the computational cost (on Amazon Review, we set $N = 20$). We set $\gamma = 0.1$ and set α following the domain adaptation methods.

Baselines. For accuracy estimation, we consider Proxy Risk (Chuang et al., 2020), Average Confidence (Avg Conf) (Elsahar and Gallé, 2019), and Ensemble Average Confidence (Ens Avg Conf). For error detection, we consider Proxy Risk, Maximum Softmax Probability (MSP) (Hendrycks and Gimpel, 2017), and Trust Score (Jiang et al., 2018). Although Proxy Risk and our method share some similar ideas such as the use of check models, disagreement and representation matching, they have some major differences in the key ideas, training objectives, and the implementation of training objectives.

6.7.2 Results

Performance for Accuracy Estimation. The results in Table 6.1 show that our method (with $\mathcal{T}_{\text{RM}}$) achieves significantly better results across various dataset categories compared to existing methods. Specifically, on Digits and CIFAR10-C, it outperforms current state-of-the-art method Proxy Risk significantly (e.g. reduce the error by $> 40\%$). On Office-31 and Amazon Review, it also outperforms the others and has a large advantage on the pre-trained models using DANN-arch.

We would like to emphasize the results on the challenging dataset iWildCam. Our method (with either $\mathcal{T}_{\text{RI}}$ or $\mathcal{T}_{\text{RM}}$) outperforms the other methods significantly (e.g., reduce the error by $> 70\%$), and the instantiation with $\mathcal{T}_{\text{RI}}$ gets the best results. Note that on iWildCam, the label distribution of the test data is imbalanced and different from that of the training data. In such a case, the representation matching technique will fail since the representations of the two domains may be misaligned. Thus, the performance of Proxy Risk becomes worse, as it relies on representation matching. Our method with $\mathcal{T}_{\text{RM}}$ also uses DANN in the ensemble method, but performs significantly better than Proxy Risk, since the diversity helps satisfy condition (**C**) though the representation matching fails to improve accuracy there. This shows that our ensemble and self-training techniques could alleviate the drawbacks of representation matching in such cases. Furthermore, our method with $\mathcal{T}_{\text{RI}}$ achieves even better results, which demonstrates the flexibility and effectiveness of our framework.

Task	Accuracy Estimation			Error Detection		
Metric	Absolute Estimation Error ↓			F1 score ↑		
Dataset	Method	Model f		Method	Model f	
		Typical DNN	DANN-arch		Typical DNN	DANN-arch
Digits	Avg Conf	0.404±0.180	0.350±0.230	MSP	0.467±0.195	0.485±0.209
	Ens Avg Conf	0.337±0.229	0.246±0.230	Trust Score	0.496±0.195	0.484±0.187
	Proxy Risk	0.085±0.142	0.095±0.181	Proxy Risk	0.844±0.118	0.796±0.155
	Ours (RI)	0.164±0.218	0.087±0.077	Ours (RI)	0.698±0.235	0.701±0.126
	Ours (RM)	**0.023**±0.020	**0.024**±0.022	Ours (RM)	**0.881**±0.084	**0.841**±0.112
Office-31	Avg Conf	0.054±0.044	0.259±0.134	MSP	0.281±0.266	0.584±0.128
	Ens Avg Conf	0.080±0.041	0.281±0.136	Trust Score	0.401±0.240	0.559±0.143
	Proxy Risk	0.033±0.012	0.042±0.034	Proxy Risk	0.605±0.177	0.629±0.140
	Ours (RI)	0.051±0.038	0.044±0.031	Ours (RI)	0.715±0.124	0.770±0.027
	Ours (RM)	**0.029**±0.021	**0.018**±0.023	Ours (RM)	**0.767**±0.052	**0.790**±0.087
CIFAR10-C	Avg Conf	0.353±0.175	0.369±0.176	MSP	0.505±0.043	0.550±0.043
	Ens Avg Conf	0.237±0.144	0.237±0.133	Trust Score	0.494±0.045	0.568±0.060
	Proxy Risk	0.053±0.070	0.052±0.070	Proxy Risk	0.850±0.107	0.843±0.101
	Ours (RI)	0.149±0.089	0.197±0.115	Ours (RI)	0.654±0.064	0.568±0.063
	Ours (RM)	**0.022**±0.009	**0.029**±0.012	Ours (RM)	**0.872**±0.083	**0.860**±0.091
iWildCam	Avg Conf	0.388±0.045	0.395±0.043	MSP	0.692±0.006	0.741±0.009
	Ens Avg Conf	0.177±0.025	0.158±0.020	Trust Score	0.717±0.009	0.737±0.010
	Proxy Risk	0.119±0.043	0.094±0.036	Proxy Risk	0.755±0.038	0.773±0.039
	Ours (RI)	**0.015**±0.008	**0.007**±0.004	Ours (RI)	0.792±0.013	0.806±0.012
	Ours (RM)	0.035±0.022	0.026±0.024	Ours (RM)	**0.796**±0.014	**0.809**±0.010
Amazon Review	Avg Conf	0.290±0.043	0.310±0.045	MSP	0.420±0.022	0.218±0.031
	Ens Avg Conf	0.229±0.038	0.217±0.036	Trust Score	0.414±0.024	0.237±0.044
	Proxy Risk	0.021±0.014	0.037±0.076	Proxy Risk	0.417±0.042	0.434±0.046
	Ours (RI)	0.065±0.037	0.062±0.051	Ours (RI)	0.384±0.032	**0.453**±0.037
	Ours (RM)	**0.018**±0.010	**0.022**±0.011	Ours (RM)	**0.426**±0.036	0.440±0.037

Table 6.1: Results for unsupervised accuracy estimation and error detection. For typical DNN, We use CNN-BN for Digits, ResNet50 for Office-31, ResNet34 for CIFAR10-C, ResNet50 for iWildCam, and Fully Connected Network for Amazon Review. We show the mean and standard deviation of absolute estimation error and F1 score (mean±std). The numbers are calculated over the training-test dataset pairs in each dataset category. **Bold** numbers are the superior results.

The results for each dataset pair are plotted in Figure 6.1. The plots show that proxy risk tends to underestimate the accuracy. This is because proxy risk maximizes disagreement which can overly suppress the accuracy. While the average confidence methods tend to overestimate the accuracy, because the model f tends to be overconfident on the data with dataset shift, even when this issue gets rectified in the ensemble average confidence method. In comparison, our method with $\mathcal{T}_{\text{RM}}$ exhibits a clear advantage.

Performance for Error Detection. Table 6.1 shows that our method with $\mathcal{T}_{\text{RM}}$ outperforms existing methods on all dataset categories. Specifically, our method improves the F1 score by at least 4.4% on Digits, by at least 25.6% on Office-31, by at least 2.0% on CIFAR10-C, by at least 4.7% on iWildCam and by at least 1.4% on Amazon Review. This shows the advantages of our method to identify error points in the unlabeled test dataset.

Ablation Studies. To study the effect of using ensembles, we vary the ensemble size N ($N = 1$ means one single h, without ensemble) in our method with $\mathcal{T}_{\text{RM}}$. Similarly, for self-training, we vary the self-training iteration number T ($T = 1$ means no self-training). Figure 6.2 shows their effect on the average F1 score: both ensemble and self-training techniques have positive effects for identifying error points and thus also for accuracy estimation. Moreover, increasing T and N can lead to further improvement. In addition to N and T, we similarly exam the effect of the last hyper-parameter γ. The figure shows that a wide range of γ can lead to good results, so it is easy to pick a suitable γ.

Validating the Theoretical Analysis. Our analysis relies on three conditions **(A)(B)(C)** stated in Section 6.5. The experimental results in Table 6.2 show that empirically they are roughly satisfied. For example, for typical DNN f on MNIST→MNIST-M, $\tilde{\nu} = 3.39\%$, $\tilde{\gamma} = 0.73\%$ while $\sigma_L^2 = 26.58\%$. We note that our theoretical analysis is for formalizing our intuition and is for the worst case. Even

Dataset Category	Digits			Office-31		
Dataset Pair	M→MM	M→U	MM→U	A→D	D→W	A→W
Actual Acc	27.19	67.56	60.04	76.31	95.97	72.96
Estimated Acc w/o self-training	33.05	70.30	69.76	80.52	95.85	74.34
Estimated Acc w/ self-training	27.50	68.21	64.28	78.11	95.09	70.57
$\tilde{\nu}$	3.15	2.18	4.22	6.16	2.17	11.11
$\tilde{\gamma}$	0.57	0.90	3.82	1.92	0.20	0.29
σ_L^2	26.54	12.89	24.57	15.61	8.80	14.67

Table 6.2: Empirical results to support the theoretical analysis. We use typical DNN as the architecture for the model f. M is MNIST, MM is MNIST-M, U is USPS, A is Amazon, D is Dslr and W is Webcam. The ensemble training algorithm we use is $\mathcal{T}_{RM}$. On Digits, we use $N = 10$ and $T = 3$ while on Office-31, we use $N = 15$ and $T = 2$. All values are percentages.

when our assumptions are not fully satisfied on some complex datasets, our method can still have empirical performance better than the error bound.

6.8 Discussions

While our framework is general and flexible when combined with different ensemble methods, and it is easy to design ensemble methods satisfying the success conditions, we note that some prior knowledge about the data is needed to achieve the best performance. For example, iWildCam has imbalanced classes which hurt representation matching and thus $\mathcal{T}_{RM}$, while $\mathcal{T}_{RI}$ is more suitable for such data. Different instantiations thus have their own limitations. For $\mathcal{T}_{RM}$, matching failure can cause errors on f's correct points, and too strong matching can decrease the diversity since the models are too restricted. For $\mathcal{T}_{RI}$, it does not attempt to improve the accuracy on the test inputs and relies only on diversity to satisfy condition (**C**). Then it can have worse performance than $\mathcal{T}_{RM}$ on test data with connection to the training data that can be exploited. The success conditions we identified can guide the design of different instantiations. We focus on ensembles of deep learning models, since they readily satisfy the conditions. However, other ensemble methods with other types of models may also be useful. We leave the exploration for future work.

6.9 Conclusion

In this chapter, we proposed a principled and practically effective framework that simultaneously addressed two challenging tasks – *unsupervised accuracy estimation* and *error detection*. Our theoretical analysis demonstrated that our framework enjoyed provable guarantees for both accuracy estimation and error detection under mild conditions readily satisfied by practical deep learning models. Along with the framework, we proposed and experimented with two instantiations and achieved state-of-the-art results on 59 tasks.

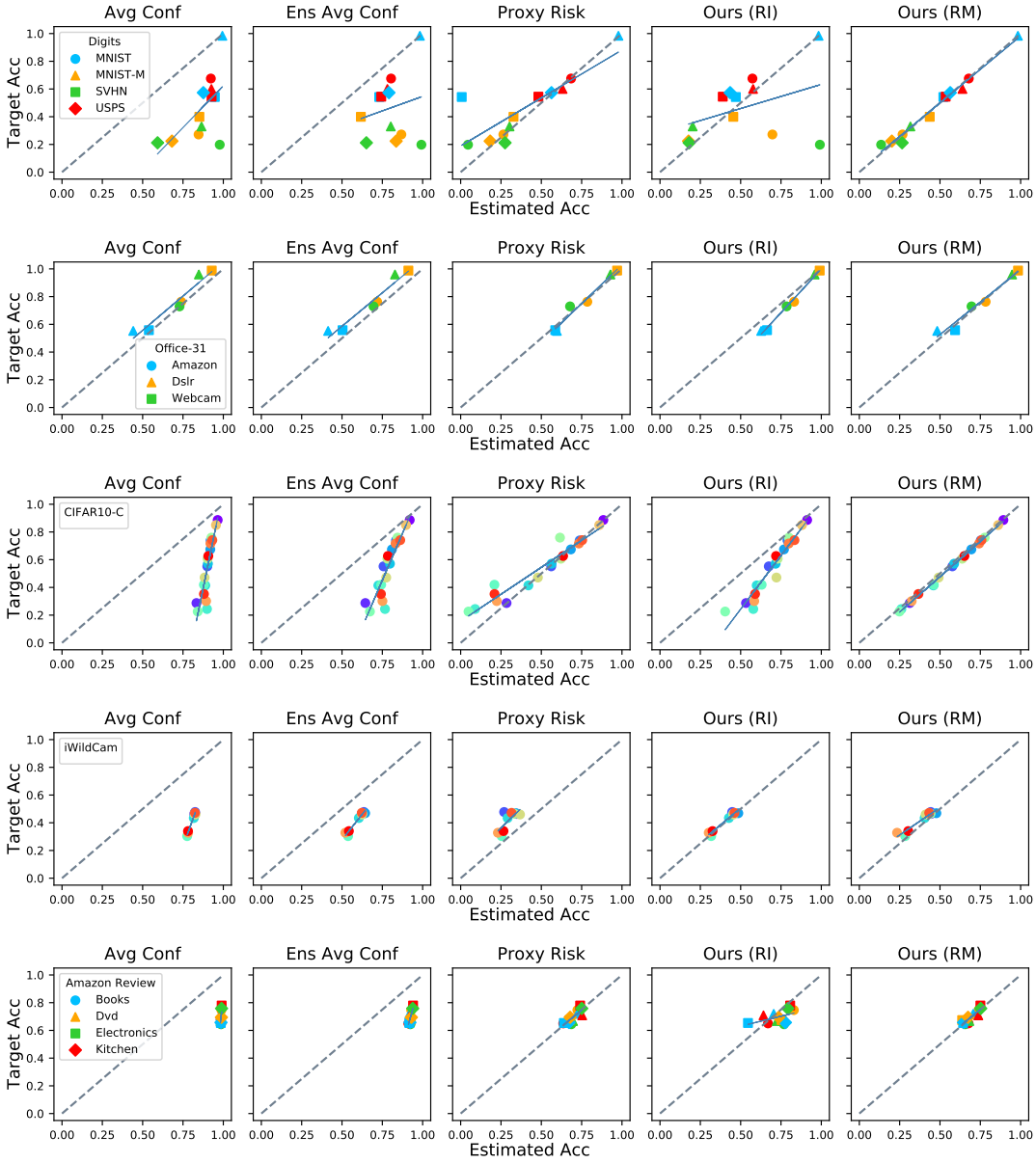


Figure 6.1: Accuracy estimation detailed results for each dataset pair. We use typical DNN as the architecture for the model f. We use symbols to represent training datasets and colors to represent test datasets. For CIFAR10-C, there is only one training dataset with multiple test datasets. The dashed line represents perfect prediction (target accuracy = estimated accuracy). Points beneath (above) the dashed line indicate overestimation (underestimation). The solid lines are regression lines of the results.

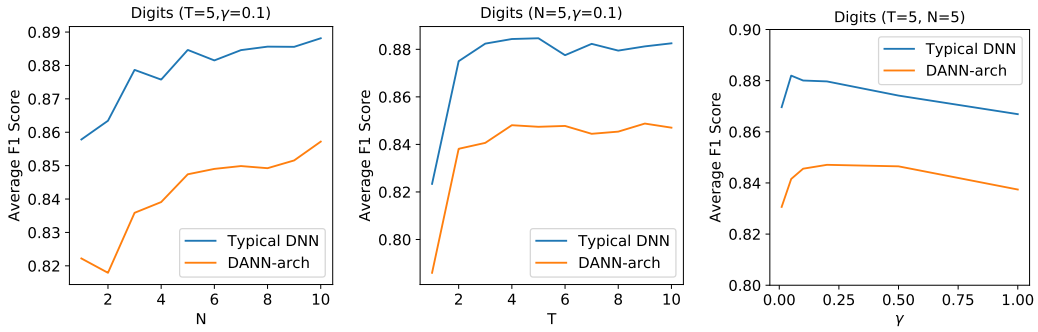


Figure 6.2: Ablation study for the effect of ensemble and self-training techniques on Digits. N is the number of models in the ensemble, T is the number of self-training iterations, and γ is the weighting parameter for the loss term on the pseudo-labeled data. The ensemble training algorithm we use is $\mathcal{T}_{\text{RM}}$.

7 ASPEST: BRIDGING THE GAP BETWEEN ACTIVE LEARNING AND SELECTIVE PREDICTION

Contribution statement. This chapter is joint work with Jinsung Yoon, Sayna Ebrahimi, Sercan O Arik, Somesh Jha, and Tomas Pfister. The author Jiefeng Chen proposed the method and completed all the experiments.

7.1 Introduction

Deep Neural Networks (DNNs) have shown notable success in many applications that require complex understanding of input data (He et al., 2016a; Devlin et al., 2019; Hannun et al., 2014), including the ones that involve high-stakes decision making (Yang, 2020). For safe deployment of DNNs in high-stakes applications, it is typically required to allow them to abstain from their predictions that are likely to be wrong, and ask humans for assistance (a task known as selective prediction) (El-Yaniv et al., 2010; Geifman and El-Yaniv, 2017). Although selective prediction can render the predictions more reliable, it does so at the cost of human interventions. For example, if a model achieves 80% accuracy on the test data, an ideal selective prediction algorithm should reject those 20% misclassified samples and send them to a human for review.

Distribution shift can significantly exacerbate the need for such human intervention. The success of DNNs often relies on the assumption that both training and test data are sampled independently and identically from the same distribution. In practice, this assumption may not hold and can degrade the performance on the test domain (Barbu et al., 2019; Koh et al., 2021). For example, for satellite imaging applications, images taken in different years can vary drastically due to weather, light, and climate conditions (Koh et al., 2021). Existing selective prediction methods usually rely on model confidence to reject inputs (Geifman and El-Yaniv, 2017). However, it has been observed that model confidence can be poorly calibrated, especially with distribution shifts (Ovadia et al., 2019). The selective

classifier might end up accepting many mis-classified test inputs, making the predictions unreliable. Thus, selective prediction might yield an accuracy below the desired target performance, or obtain a low coverage, necessitating significant human intervention.

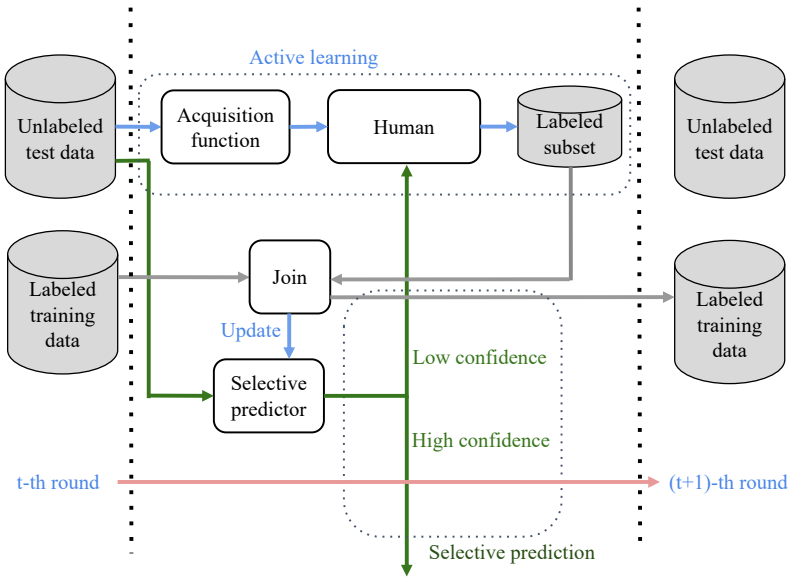


Figure 7.1: Illustration of the *active selective prediction* problem, where active learning is used to improve selective prediction under distribution shift. In this setting, active learning selects a small subset of data for labeling which are used to improve selective prediction on the remaining unlabeled test data. This yields more reliable predictions and more optimized use of humans in the loop.

To improve the performance of selective prediction, one idea is to rely on active learning and to have humans label a small subset of selected test data. The correct labels provided by humans can then be used to improve the accuracy and coverage (see Sec. 7.3.2) of selective prediction on the remaining unlabeled test data, thus reducing the need for subsequent human labeling efforts. In separate forms, selective prediction (Geifman and El-Yaniv, 2017; Geifman and El-Yaniv, 2019) and active learning (Settles, 2009) have been studied extensively, however, to the best of our knowledge, we are the first to propose performing active learning to

improve selective prediction jointly, with the focus on the major real-world challenge of distribution shifts. Active domain adaptation (Su et al., 2020; Fu et al., 2021; Prabhu et al., 2021) is one area close to this setting, however, it does not consider selective prediction. In selective prediction, not only does a classifier need to be learned, but a selection scoring function also needs to be constructed for rejecting misclassified inputs. Thus, going beyond conventional active learning methods that focus on selecting examples for labeling to improve the accuracy, we propose to also use those selected labeled examples to improve the selection scoring function. The optimal acquisition function (used to select examples for labeling) for this new setting is different compared to those in traditional active learning – e.g. if a confidence-based selection scoring function is employed, the selected labeled samples should have the goal of improving the estimation of that confidence score.

In this chapter, we introduce a new machine learning paradigm: active selective prediction under distribution shift (see Fig. 7.1), which combines selective prediction and active learning to improve accuracy and coverage, and hence use human labeling in a more optimal way. Active selective prediction is highly important for most real-world deployment scenarios. To the best of our knowledge, we are the first to formulate and investigate this problem, along with the judiciously chosen evaluation metrics for it (Sec. 7.3). We also introduce a novel and simple yet effective method, ASPEST, for this active selective prediction problem (Sec. 7.4). The key components of ASPEST, checkpoint ensembling and self-training, are designed to address the fundamental challenges in the active selective prediction problem. On numerous real-world datasets, we show that ASPEST consistently outperforms other baselines proposed for active learning and selective prediction (Sec. 7.5).

7.2 Related Work

Selective prediction. Selective prediction (also known as prediction with rejection/deferral options) constitutes a common deployment scenario for DNNs, especially in high-stakes decision making scenarios. In selective prediction, models abstain from yielding outputs if their confidence on the likelihood of correctness

is not sufficiently high. Such abstinence usually incurs deferrals to humans and results in additional cost (Mozannar and Sontag, 2020). Increasing the coverage – the ratio of the samples for which the DNN outputs can be reliable – is the fundamental goal (El-Yaniv et al., 2010; Fumera and Roli, 2002; Hellman, 1970; Geifman and El-Yaniv, 2019). Geifman and El-Yaniv consider selective prediction for DNNs with the 'Softmax Response' method, which applies a carefully selected threshold on the maximal response of the softmax layer to construct the selective classifier. Lakshminarayanan et al. show that using deep ensembles can improve predictive uncertainty estimates and thus improve selective prediction. Rabanser et al. propose a novel method, NNTD, for selective prediction that utilizes DNN training dynamics by using checkpoints during training. Our proposed method ASPEST also uses checkpoints to construct ensembles for selective prediction. In contrast to NNTD and other aforementioned methods, we combine selective prediction with active learning to improve its data efficiency while considering a holistic perspective of having humans in the loop. This new active selective prediction setup warrants new methods for selective prediction along with active learning.

Active learning. To utilize the human labeling budget more effectively while training DNNs, active learning employs acquisition functions to select unlabeled examples for labeling, and uses these labeled examples to train models (Settles, 2009; Dasgupta, 2011). Commonly-used active learning methods employ acquisition functions by considering uncertainty (Gal et al., 2017; Ducoffe and Precioso, 2018; Beluch et al., 2018) or diversity (Sener and Savarese, 2017; Sinha et al., 2019), or their combination (Ash et al., 2019; Huang et al., 2010). One core challenge for active learning is the "cold start" problem: often the improved obtained from active learning is less significant when the amount of labeled data is significantly smaller (Yuan et al., 2020; Hacohen et al., 2022). Moreover, active learning can be particularly challenging under distribution shift (Kirsch et al., 2021; Zhao et al., 2021). Recently, active domain adaptation has been studied, where domain adaptation is combined with active learning (Su et al., 2020; Fu et al., 2021; Prabhu et al., 2021). Different from traditional active learning, active domain adaptation typically

adapts a model pre-trained on the labeled source domain to the unlabeled target domain. Although we also try to adapt a source trained model to the unlabeled target test set using active learning, we focus on building a selective classification model and reducing the human labeling effort.

Distribution shift. Distribution shift, where the training distribution differs from the test distribution, often occurs in practice and can substantially degrade the accuracy of the deployed DNNs (Koh et al., 2021; Yao et al., 2022; Barbu et al., 2019). Distribution shift can also substantially reduce the quality of uncertainty estimation (Ovadia et al., 2019), which is often used for rejecting examples in selective prediction and selecting samples for labeling in active learning. Several techniques try to tackle the challenge caused by distribution shift, including accuracy estimation (Chen et al., 2021b; Chuang et al., 2020), error detection (Hendrycks and Gimpel, 2017; Granese et al., 2021), out-of-distribution detection (Salehi et al., 2021), domain adaptation (Ganin et al., 2016; Saito et al., 2019), selective prediction (Kamath et al., 2020) and active learning (Kirsch et al., 2021). In this chapter, we combine selective prediction with active learning to address the issue of distribution shift.

Deep ensembles. Ensembles of DNNs (or deep ensembles) have been successfully used to boost predictive performance (Moghimi et al., 2016; Zhu et al., 2018). Deep ensembles can also be used to improve the predictive uncertainty estimation (Lakshminarayanan et al., 2017; Fort et al., 2019). Lakshminarayanan et al. show that random initialization of the NN parameters along with random shuffling of the data points are sufficient for deep ensembles to perform well in practice. However, training multiple DNNs from random initialization can be very expensive. To obtain deep ensembles more efficiently, recent papers explore using checkpoints during training to construct the ensemble (Wang et al., 2021b; Huang et al., 2017a), or fine-tuning a single pre-trained model to create the ensemble (Kobayashi et al., 2022). In the proposed method, we use the checkpoints during fine-tuning a source-trained model via active learning as the ensemble and further boost the ensemble's

performance via self-training. We also use the ensemble's uncertainty measured by a margin to select samples for labeling in active learning.

Self-training. Self-training is a common algorithmic paradigm for leveraging unlabeled data with DNNs. Self-training methods train a model to fit pseudo-labels (i.e., predictions on unlabeled data made by a previously-learned model) to boost the model's performance (Yarowsky, 1995; Grandvalet and Bengio, 2004; Lee et al., 2013; Wei et al., 2020; Sohn et al., 2020). In this chapter, we use self-training to improve selective prediction performance. Instead of using predicted labels as pseudo-labels as a common practice in prior works, we use the average softmax outputs of the checkpoints during training as the pseudo-labels and self-train the models in the ensemble on them with the KL-Divergence loss to improve selective prediction performance.

7.3 Active Selective Prediction

In this section, we first formulate the active selective prediction problem and then present the proposed evaluation metrics to quantify the efficacy of the methods.

7.3.1 Problem Setup

Let $\mathcal{X}$ be the input space and $\mathcal{Y} = \{1, 2, \ldots, K\}$ the label space.[1] The training data distribution is given as $P_{X,Y}$ and the test data distribution is $Q_{X,Y}$ (both are defined in the space $\mathcal{X} \times \mathcal{Y}$). There might exist distribution shifts such as covariate shifts (i.e., $Q_{X,Y}$ might be different from $P_{X,Y}$). Suppose for each input $\mathbf{x}$, an oracle (e.g., the human annotator) can assign a ground-truth class label y_x to it. Given a classifier $\bar{f} : \mathcal{X} \to \mathcal{Y}$ trained on a source training dataset $\mathcal{D}^{tr} \sim P_{X,Y}$ ($\sim$ means "sampled from"), and an unlabeled target test dataset $U_X = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\} \sim Q_X$, our goal is to employ $\bar{f}$ to yield reliable predictions on U_X in human-in-the-loop scenario. Holistically, we

[1] In this chapter, we focus on the classification problem, although it can be extended to the regression problem.

consider the two approaches to involve humans via the predictions they provide on the data: (i) selective prediction where uncertain predictions are deferred to humans to maintain a certain accuracy target; and (ii) active learning where a subset of U_X unlabeled samples are selected for humans to improve the model with the extra labeled data to be used at the subsequent iterations. These two approaches to involve humans have different objectives and thus, their joint optimization to best use the human labeling resources is not straightforward.

As an extension of the classifier f (initialized by $\bar{f}$), we propose to employ a selective classifier f_s including a selection scoring function $g : \mathcal{X} \to \mathbb{R}$ to yield reliable predictions on U_X. We define the predicted probability of the model f on the k-th class as $f(\mathbf{x} \mid k)$. Then, the classifier is $f(\mathbf{x}) = \arg\max_{k \in \mathcal{Y}} f(\mathbf{x} \mid k)$. g can be based on statistical operations on the outputs of f (e.g., $g(\mathbf{x}) = \max_{k \in \mathcal{Y}} f(\mathbf{x} \mid k)$). With f and g, the selective prediction model f_s is defined as:

$$f_s(\mathbf{x}; \tau) = \begin{cases} f(\mathbf{x}) & \text{if } g(\mathbf{x}) \geqslant \tau, \\ \perp & \text{if } g(\mathbf{x}) < \tau \end{cases}, \tag{7.1}$$

where τ is a threshold. If $f_s(\mathbf{x}) = \perp$, then the DNN system would defer the predictions to a human in the loop. To improve the overall accuracy to reach the target, such deferrals require manual labeling. To reduce the human labeling cost and improve the accuracy of the selective classifier, we consider labeling a small subset of U_X and adapt the selective classifier f_s on the labeled subset via active learning. The goal is to significantly improve the accuracy and coverage of the selective classifier f_s and thus reduce the total human labeling effort.

Suppose the labeling budget for active learning is M (i.e., M examples are selected to be labeled from U_X to improve the selective prediction performance). We assume that the human in the loop can provide the correct labels. For active learning, we consider the transductive learning paradigm (Vapnik, 1998), which assumes all training and test data are observed beforehand and we can make use of the unlabeled test data for learning. Specifically, the active learning is performed on U_X to build the selective classifier f_s, with performance evaluation of f_s only on U_X.

We don't consider training f_s from scratch, but adapt the source-trained classifier $\bar{f}$ to obtain f_s to maintain feasibly-low computational cost (e.g., by fine-tuning $\bar{f}$ on the M labeled data points from U_X).

Let's first consider the single-round setting. Suppose the acquisition function is $a : \mathcal{X}^m \times \mathcal{F} \times \mathcal{G} \to \mathbb{R}$, where $m \in \mathbb{N}^+$, $\mathcal{F}$ is the classifier space and $\mathcal{G}$ is the selection scoring function space. This acquisition function is the same as the one used in active learning literature (Gal et al., 2017) (refer to Appendix E.1 for some examples of the function a). In the beginning, f is initialized by $\bar{f}$. We then select a batch B^* for labeling by solving the following objective:

$$B^* = \arg \max_{B \subset U_X, |B|=M} a(B, f, g), \tag{7.2}$$

for which the labels are obtained to get $\tilde{B}^*$. Then, we use $\tilde{B}^*$ to update f and g (e.g., via fine-tuning the model on $\tilde{B}^*$).

The above can be extended to a multi-round setting. Suppose we have T rounds and the labeling budget for each round is $m = [\frac{M}{T}]$. In the beginning, f_0 is initialized by $\bar{f}$. At the t-th round, we first select a batch B_t^* for labeling by solving the following objective:

$$B_t^* = \arg \max_{B \subset U_X \setminus (\cup_{i=1}^{t-1} B_i^*), |B|=m} a(B, f_{t-1}, g_{t-1}), \tag{7.3}$$

for which the labels are obtained to get $\tilde{B}_t^*$. Then we use $\tilde{B}_t^*$ to update f_{t-1} and g_{t-1} to get f_t and g_t (e.g., via fine-tuning the model on $\tilde{B}_t^*$). With multiple-rounds setting, we define $B^* = \cup_{i=1}^{T} B_i^*$.

7.3.2 Evaluation Metrics

To quantify the efficacy of the methods that optimize human-in-the-loop adaptation and decision making performance, appropriate metrics are needed.

The performance of the selective classifier f_s (defined in Eq. (7.1)) is evaluated

by the accuracy and coverage metrics. The accuracy of f_s on U_X is defined as:

$$\mathrm{acc}(f_s, \tau) = \frac{\mathbb{E}_{\mathbf{x} \sim U_X} \mathbb{I}[f(\mathbf{x}) = y_x \wedge g(\mathbf{x}) \geqslant \tau \wedge \mathbf{x} \notin B^*]}{\mathbb{E}_{\mathbf{x} \sim U_X} \mathbb{I}[g(\mathbf{x}) \geqslant \tau \wedge \mathbf{x} \notin B^*]} \tag{7.4}$$

Here, the accuracy is measured on the predictions made by the model without human intervention (excluding those predictions on B^* and rejected data points). The coverage of f_s on U_X is defined as:

$$\mathrm{cov}(f_s, \tau) = \frac{\mathbb{E}_{\mathbf{x} \sim U_X} \mathbb{I}[g(\mathbf{x}) \geqslant \tau \wedge \mathbf{x} \notin B^*]}{\mathbb{E}_{\mathbf{x} \sim U_X} \mathbb{I}[\mathbf{x} \notin B^*]} \tag{7.5}$$

The coverage is the fraction of remaining unlabeled data points where we can rely on the model's prediction without human intervention. We can tune the threshold τ to achieve a certain coverage. We know that there could be an accuracy-coverage trade-off – as we increase coverage, the accuracy could be lower. We consider the following metrics that are agnostic to the threshold τ:

Maximum Accuracy at a Target Coverage. Given a target coverage t_c, the maximum accuracy is defined as:

$$\max_{\tau} \quad \mathrm{acc}(f_s, \tau), \quad \text{s.t.} \quad \mathrm{cov}(f_s, \tau) \geqslant t_c \tag{7.6}$$

We denote this metric as $\mathrm{acc}|\mathrm{cov} \geqslant t_c$.

Maximum Coverage at a Target Accuracy. Given a target accuracy t_a, the maximum coverage is defined as:

$$\max_{\tau} \quad \mathrm{cov}(f_s, \tau), \quad \text{s.t.} \quad \mathrm{acc}(f_s, \tau) \geqslant t_a \tag{7.7}$$

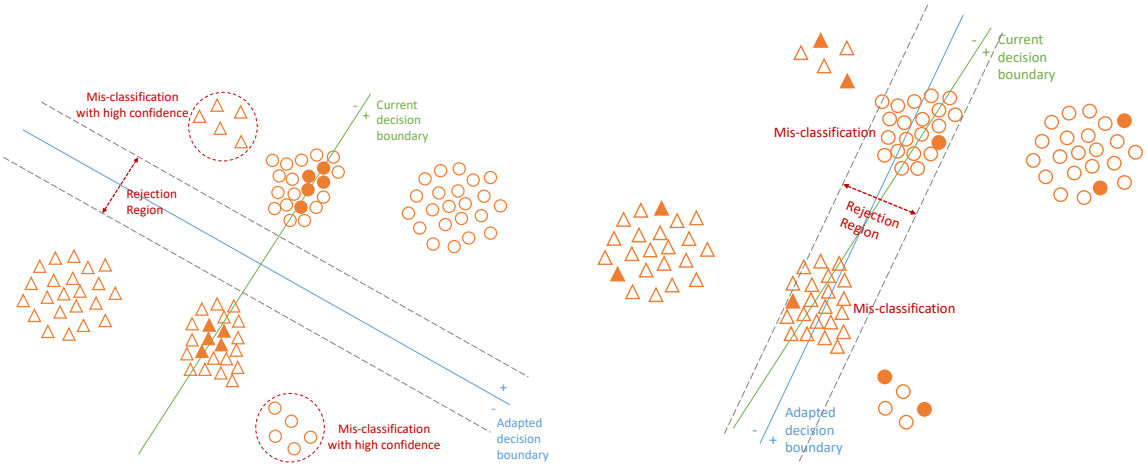
When $\tau = \infty$, we define $\mathrm{cov}(f_s, \tau) = 0$ and $\mathrm{acc}(f_s, \tau) = 1$. We denote this metric as $\mathrm{cov}|\mathrm{acc} \geqslant t_a$.

Area Under the Accuracy-Coverage Curve (AUC). We define the AUC metric as:

$$\mathrm{AUC}(f_s) = \int_0^1 \mathrm{acc}(f_s, \tau)\mathrm{dcov}(f_s, \tau) \tag{7.8}$$

We use the composite trapezoidal rule to estimate the integration.

7.3.3 Challenges



(a) Sample selection based on uncertainty (b) Sample selection based on diversity

Figure 7.2: Illustration of the challenges in active selective prediction using a linear model to maximize the margin (distance to the decision boundary) for binary classification. The model confidence is considered to be proportional to the margin (when the margin is larger, the confidence is higher and vice versa). The triangles belong to the negative class while the circles belong to the positive class. The empty markers represent the *unlabeled* test samples while the solid markers are the selected samples for labeling in active learning. Fig. (a) shows that if the samples close to the current decision boundary are selected for labeling, then the adapted model suffers from the overconfidence issue (mis-classification with high confidence), which results in acceptance of some mis-classified points. Fig. (b) shows that if diverse samples are selected for labeling, then the adapted model suffers from low accuracy. This leads to rejection of many points, necessitating significant human intervention.

For active selective prediction, we want to utilize active learning to improve the coverage and accuracy of the selective classifier f_s, that consists of a classifier f and a selection scoring function g. Different from conventional active learning, which only aims to improve the accuracy of the classifier f, active selective prediction also aims to improve g so that it can accept those examples where f predicts correctly and reject those where f predicts incorrectly. With distribution shift and a small labeling budget M, it can be challenging to train f for high accuracy. Therefore, g is critical in achieving high coverage and accuracy of f_s, for which we consider the confidence of f (i.e., the maximum softmax score of f) and train f such that its confidence can be used to distinguish correct and incorrect predictions. This might not be achieved easily since it has been observed that under distribution shift, f can have overconfident predictions (Goodfellow et al., 2015; Hein et al., 2019). Besides, for active learning, typically we select samples for labeling based on uncertainty or diversity. However, in active selective prediction, sample selection based on uncertainty may lead to the overconfidence issue and sample selection based on diversity may lead to low accuracy of f, as illustrated in Fig. 7.2. The results in Table 7.1 show that these issues indeed exist – the methods based on uncertainty sampling (SR+Confidence, SR+Entropy and SR+Margin) achieve relatively high accuracy of f, but suffer from the overconfidence issue (i.e., mis-classification with high confidence). The method based on diversity sampling (SR+kCG) doesn't have the overconfidence issue, but suffers from low accuracy of f. Also, the hybrid methods based on uncertainty and diversity sampling (SR+CLUE and SR+BADGE) still suffer from the overconfidence issue. To tackle these, we propose a novel method ASPEST. The results show that the proposed method ASPEST achieves much higher accuracy of f, effectively alleviates the overconfidence issue, and significantly improves the selective prediction performance. We describe ASPEST next.

7.4 Proposed Method: ASPEST

We propose a novel method called Active Selective Prediction using Ensem-

Method	Accuracy of f ↑	Overconfidence ratio ↓	AUC↑
SR+Confidence	45.29±3.39	16.91±2.24	64.14±2.83
SR+Entropy	45.78±6.36	36.84±18.96	65.88±4.74
SR+Margin	58.10±0.55	13.18±1.85	76.79±0.45
SR+kCG	32.68±3.87	**0.04**±0.01	48.83±7.21
SR+CLUE	55.22±2.27	9.47±0.94	73.15±2.68
SR+BADGE	56.55±1.62	8.37±2.56	76.06±1.63
ASPEST (ours)	**71.82**±1.49	0.10±0.02	**88.84**±1.02

Table 7.1: Evaluating the Softmax Response (SR) method with various active learning methods and the proposed ASPEST on MNIST→SVHN. The experimental setup is describe in Section 7.5.1. The labeling budget M is 100. The overconfidence ratio is the ratio of *mis-classified* unlabeled test inputs that have confidence $\geqslant 1$ (the highest confidence). The mean and std of each metric over three random runs are reported (mean±std). All numbers are percentages. **Bold** numbers are superior results.

bles and Self-training (ASPEST), which utilizes two key techniques checkpoint ensembles and self-training, to solve the active selective prediction problem. The key constituents, checkpoint ensembles and self-training, are designed to tackle the fundamental challenges in active selective prediction, with the ideas of selecting samples for labeling based on uncertainty to achieve high accuracy and using checkpoint ensembles and self-training to alleviate the overconfidence issue. We empirically analyze why they can tackle the challenges in Section 7.5.3. The full ASPEST algorithm is shown in Algorithm 4 and the details are explained next.

We first describe how the weigths from the intermediate model checkpoints during training are used to construct the checkpoint ensemble. Since we have all the test inputs, we don't need to save the checkpoints during training, but just record their outputs on the test set U_X. Specifically, we use a $n \times K$ matrix P (recall that $n = |U_X|$ and K is the number of classes) to record the average of the softmax outputs of the checkpoint ensemble and use N_e to record the number of checkpoints in the current checkpoint ensemble. During training, we get a stream of checkpoints,

Algorithm 4 Active Selective Prediction using Ensembles and Self-Training

Require: A training set $\mathcal{D}^{\text{tr}}$, a unlabeled test set U_X, the number of rounds T, the labeling budget M, the number of models N, the number of initial training steps n_s, the initial checkpoint steps c_s, a checkpoint epoch c_e, a threshold η, a sub-sampling fraction p, and a hyper-parameter λ.
Let $f_0^j = \bar{f}$ for $j = 1, \ldots, N$.
Set $N_e = 0$ and $P = \mathbf{0}_{n \times K}$.
Fine-tune each f_0^j for n_s training steps using objective (7.10) and update P and N_e using Eq. (7.9) every c_s training steps.
for $t = 1, \cdots, T$ **do**
  Select a batch B_t from U_X for labeling using the sample selection objective (7.11).
  Use an oracle to assign ground-truth labels to the examples in B_t to get $\tilde{B}_t$.
  Set $N_e = 0$ and $P = \mathbf{0}_{n \times K}$.
  Fine-tune each f_{t-1}^j using objective (7.12), while updating P and N_e using Eq (7.9) every c_e training epochs.
  Let $f_t^j = f_{t-1}^j$.
  Construct the pseudo-labeled set R via Eq (7.13) and create R_{sub} by randomly sampling up to $\lceil p \cdot n \rceil$ data points from R.
  Train each f_t^j further via SGD using the objective (7.14) and update P and N_e using Eq (7.9) every c_e training epochs.
end for

Ensure: The classifier $f(\mathbf{x}_i) = \arg\max_{k \in \mathcal{Y}} P_{i,k}$ and the selection scoring function $g(\mathbf{x}_i) = \max_{k \in \mathcal{Y}} P_{i,k}$.

and for each incoming checkpoint model f, we update P and N_e as:

$$P_{i,k} \leftarrow \frac{1}{N_e + 1}(P_{i,k} \cdot N_e + f(\mathbf{x}_i \mid k)) \quad \text{for } 1 \leqslant i \leqslant n \text{ and } 1 \leqslant k \leqslant K, \tag{7.9}$$
$$N_e \leftarrow N_e + 1.$$

Since it has been observed that an ensemble of DNNs (known as 'deep ensembles') usually produces a confidence score that is better calibrated compared to a single DNN (Lakshminarayanan et al., 2017), we consider f to be in the form of deep ensembles and g to be the confidence of the ensemble. Specifically, we continue

fine-tuning N models independently via Stochastic Gradient Descent (SGD) with different random seeds (e.g., the randomness can come from different random orders of training batches). At the beginning, we set each model $f_0^j = \bar{f}$ ($j = 1, \ldots, N$), and set $N_e = 0$ and $P = \mathbf{0}_{n \times K}$. Here, we initialize each model f_0^j with the source-trained classifier $\bar{f}$ instead of random initialization, to minimize the computational cost. We fine-tune each model f_0^j on $\mathcal{D}^{\text{tr}}$ for n_s steps via SGD using the following training objective:

$$\min_{\theta^j} \quad \mathbb{E}_{(\mathbf{x},y)\in\mathcal{D}^{\text{tr}}} \quad \ell_{CE}(\mathbf{x}, y; \theta^j), \tag{7.10}$$

where ℓ_{CE} is the cross-entropy loss and θ^j is the model parameters of f_0^j. For every c_s steps when training each f_0^j, we update P and N_e using Eq (7.9) with the checkpoint model f_0^j.

After constructing the initial checkpoint ensemble, we perform a T-round active learning process. In each round of active learning, we first select samples for labeling based on the margin of the checkpoint ensemble, then fine-tune the models on the selected labeled test data, and finally perform self-training. We describe the procedure below:

Sample selection. In the t-th round, we select a batch B_t with a size of $m = [\frac{M}{T}]$ from U_X via:

$$B_t = \arg \max_{B \subset U_X \setminus (\cup_{l=0}^{t-1} B_l), |B|=m} - \sum_{\mathbf{x}_i \in B} S(\mathbf{x}_i) \tag{7.11}$$

where $B_0 = \emptyset$, $S(\mathbf{x}_i) = P_{i,\hat{y}} - \max_{k \in \mathcal{Y} \setminus \{\hat{y}\}} P_{i,k}$ and $\hat{y} = \arg\max_{k \in \mathcal{Y}} P_{i,k}$. We use an oracle to assign ground-truth labels to the examples in B_t to get $\tilde{B}_t$. Here, we select the test samples for labeling based on the margin of the checkpoint ensemble. The test samples with lower margin should be closer to the decision boundary and they are data points where the ensemble is uncertain about its predictions. Training on those data points can either make the predictions of the ensemble more accurate or make the ensemble have higher confidence on its correct predictions.

Fine-tuning. After the sample selection, we reset N_e and P as $N_e = 0$ and $P = \mathbf{0}_{n \times K}$, because we want to remove those checkpoints in the previous rounds with a worse performance from the checkpoint ensemble. We then fine-tune each model f_{t-1}^j $(j = 1, \ldots, N)$ independently via SGD with different randomness on the selected labeled test data to get f_t^j using the following training objective:

$$\min_{\theta^j} \quad \mathbb{E}_{(\mathbf{x},y) \in \cup_{l=1}^{t} \tilde{B}_l} \quad \ell_{CE}(\mathbf{x}, y; \theta^j) + \lambda \cdot \mathbb{E}_{(\mathbf{x},y) \in \mathcal{D}^{tr}} \quad \ell_{CE}(\mathbf{x}, y; \theta^j), \tag{7.12}$$

where θ^j is the model parameters of f_{t-1}^j and λ is a hyper-parameter. Note that here we use joint training on $\mathcal{D}^{tr}$ and $\cup_{l=1}^{t} \tilde{B}_l$ to avoid over-fitting to the small set of labeled test data and prevent the models from forgetting the source training knowledge. For every c_e epoch when fine-tuning each model f_{t-1}^j, we update P and N_e using Eq. (7.9) with the checkpoint model f_{t-1}^j.

Self-training. After fine-tuning the models on the selected labeled test data and with the checkpoint ensemble, we construct a pseudo-labeled set R via:

$$R = \{(\mathbf{x}_i, P_{i,:}) \mid \mathbf{x}_i \in U_X \wedge (\eta \leqslant \max_{k \in \mathcal{Y}} P_{i,k} < 1)\}, \tag{7.13}$$

where $\max_{k \in \mathcal{Y}} P_{i,k}$ is the confidence of the checkpoint ensemble on $\mathbf{x}_i$ and η is a threshold. We do not add those test data points with confidence equal to 1 into the pseudo-labeled set because training on those data points cannot change the models much and may even hurt the performance. We then perform self-training on the pseudo-labeled set R. For computational efficiency, we only apply self-training on a subset of R. We construct the subset R_{sub} by randomly sampling up to $[p \cdot n]$ data points from R, where $p \in [0, 1]$. We train each model f_t^j $(j = 1, \ldots, N)$ further on the pseudo-labeled subset R_{sub} via SGD using the following training objective:

$$\min_{\theta^j} \quad \mathbb{E}_{(\mathbf{x},\mathbf{y}) \in R_{sub}} \quad \ell_{KL}(\mathbf{x}, \mathbf{y}; \theta^j) + \lambda \cdot \mathbb{E}_{(\mathbf{x},y) \in \mathcal{D}^{tr}} \quad \ell_{CE}(\mathbf{x}, y; \theta^j) \tag{7.14}$$

where ℓ_{KL} is the KL-Divergence loss, which is defined as: $\ell_{KL}(\mathbf{x}, \mathbf{y}; \theta) = \sum_{k=1}^{K} \mathbf{y}_k \cdot \log(\frac{\mathbf{y}_k}{f(\mathbf{x}|k;\theta)})$. Note that for self-training, we typically use predicted labels as pseudo-labels and the cross-entropy loss. We don't follow this because the predicted labels might be wrong and training the models on those misclassified pseudo-labeled data points using the cross entropy loss will make the models have very high confidence on their wrong predictions, which will hurt selective prediction performance. For every c_e epoch of self-training each model f_t^j, we will update P and N_e using Eq (7.9) with the checkpoint model f_t^j. We add checkpoints during self-training into checkpoint ensemble to improve the sample selection in the next round of active learning.

After T rounds active learning, we use the checkpoint ensemble as the final selective classifier: the classifier $f(\mathbf{x}_i) = \arg\max_{k \in \mathcal{Y}} P_{i,k}$ and the selection scoring function $g(\mathbf{x}_i) = \max_{k \in \mathcal{Y}} P_{i,k}$.

7.5 Experiments

This section presents experimental results, especially focusing on the following questions: (**Q1**) Can we use a small labeling budget to significantly improve selective prediction performance under distribution shift? (**Q2**) Does the proposed ASPEST outperform baselines across different datasets with distribution shift? (**Q3**) What is the effect of checkpoint ensembles and self-training in ASPEST?

7.5.1 Setup

Datasets. We perform experiments on the following datasets with distribution shift: (i) MNIST→SVHN (LeCun, 1998; Netzer et al., 2011), (ii) CIFAR-10→CINIC-10 (Krizhevsky et al., 2009; Darlow et al., 2018), (iii) FMoW (Koh et al., 2021), (iv) Amazon Review (Koh et al., 2021), (v) DomainNet (Peng et al., 2019) and (vi) Otto (Benjamin Bossan, 2015). Details of the datasets are described in Appendix E.2.1.

Architectures and training. On MNIST→SVHN, we use a Convolutional Neural Network (CNN) (LeCun et al., 1989) model. On CIFAR-10→CINIC-10, we use the ResNet-20 model (He et al., 2016b). On the FMoW dataset, we use the DensetNet-121 model (Huang et al., 2017b). On Amazon Review, we use the pre-trained RoBERTa model (Liu et al., 2019). On the DomainNet dataset, we use the ResNet-50 model (He et al., 2016a). On the Otto dataset, we use a multi-layer perceptron. On each dataset, we train the models on the training set $\mathcal{D}^{\text{tr}}$. More details on model architectures and training on source data are presented in Appendix E.2.2.

Active learning hyper-parameters. We evaluate different methods with different labeling budget M values on each dataset. By default, we set the number of rounds $T = 10$ for all methods. During the active learning process, we fine-tune the model on the selected labeled test data. During fine-tuning, we don't apply any data augmentation to the test data. We use the same fine-tuning hyper-parameters for different methods to ensure a fair comparison. More details on the fine-tuning hyper-parameters can be found in Appendix E.2.3.

Baselines. We consider Softmax Response (SR) (Geifman and El-Yaniv, 2017) and Deep Ensembles (DE) (Lakshminarayanan et al., 2017) with various active learning sampling methods as the baselines. SR+Uniform means combining SR with an acquisition function based on uniform sampling (similarly for DE and other acquisition functions). We consider sampling methods from both traditional active learning (e.g., BADGE (Ash et al., 2019)) and active domain adaptation (e.g., CLUE (Prabhu et al., 2021)). Appendix E.1 further describes the details of the baselines.

Hyper-parameters of ASPEST. We set $\lambda = 1$, $n_s = 1000$ and $N = 5$, which are the same as those for Deep Ensembles, for fair comparisons. For all datasets, we use $c_s = 200$, $p = 0.1$, $\eta = 0.9$, the number of self-training epochs to be 20 and $c_e = 5$. Note that we don't tune c_s, c_e, p and use the fixed values. We select η based on

the performance on a validation dataset (i.e., DomainNet R→I) and use the same value across all other datasets.

7.5.2 Results

Dataset	MNIST→SVHN	
Metric	$\text{cov}^*\|\text{acc} \geqslant 90\% \uparrow$	$\text{acc}\|\text{cov}^* \geqslant 90\% \uparrow$
SR (without active learning)	0.08±0.0	25.80±0.0
SR+Margin (M=500)	62.38±2.7	80.21±0.9
SR+Margin (M=1000)	79.04±0.2	85.36±0.3
DE (without active learning)	0.12±0.1	28.17±0.5
DE+Margin (M=500)	76.35±2.7	84.34±1.1
DE+Margin (M=1000)	89.19±0.3	89.59±0.1
ASPEST (M=500)	87.51±0.9	88.88±0.4
ASPEST (M=1000)	**94.91**±0.4	**92.44**±0.2

Table 7.2: Results on MNIST→SVHN to describe the effect of combining selective prediction with active learning. The mean and std of each metric over three random runs are reported (mean±std). cov^* is defined in Appendix E.3.3. All numbers are percentages. **Bold** numbers are superior results.

Impacts of combining selective prediction with active learning. We evaluate the accuracy of the source trained models on the test set U_X of different datasets. The results in Appendix E.3.1 show that the models trained on the source training set $\mathcal{D}^{\text{tr}}$ suffer a performance drop on the target test set U_X, and sometimes this drop can be large. For example, the model trained on MNIST has a source test accuracy of 99.40%. However, its accuracy on the target test set U_X from SVHN is only 24.68%. If we directly build a selective classifier on top of the source trained model, then to achieve a target accuracy of 90%, the coverage would be at most 27.42%. In Table 7.2, we demonstrate that for a target accuracy of 90%, the coverage achieved by SR and DE without active learning is very low (nearly 0%). It means that almost all test examples need human intervention or labeling. This is a large cost since the test set of SVHN contains over 26K images. However, by combining selective prediction with active learning (e.g., using the proposed method ASPEST), we only need to label 500 test examples to achieve a target accuracy of 90% with a

coverage of 87.5%. Thus, during active learning and selective prediction processes, only 12.5% test examples from SVHN need to be labeled by a human to achieve the target accuracy of 90%, resulting in a significant reduction of the overall human labeling cost. Similar results are observed for other datasets (see Appendix E.3.3).

Dataset	DomainNet R→C (easy)		Amazon Review		Otto	
Metric	cov|acc $\geqslant$ 80% ↑	AUC ↑	cov|acc $\geqslant$ 80% ↑	AUC ↑	cov|acc $\geqslant$ 80% ↑	AUC ↑
SR+Uniform	25.56±0.6	63.31±0.4	13.71±11.3	72.71±1.5	63.58±0.7	84.46±0.2
SR+Confidence	25.96±0.2	64.20±0.6	11.28±8.9	72.89±0.7	69.63±1.7	85.91±0.3
SR+Entropy	25.44±1.0	63.52±0.6	5.55±7.8	71.96±1.6	67.79±0.8	85.41±0.3
SR+Margin	26.28±1.2	64.37±0.8	14.48±10.9	73.25±1.0	68.10±0.1	85.56±0.1
SR+kCG	21.12±0.3	58.88±0.0	20.02±11.0	72.34±3.2	64.84±0.7	85.08±0.2
SR+CLUE	27.17±0.8	64.38±0.6	4.15±5.9	73.43±0.4	68.21±1.2	85.82±0.3
SR+BADGE	27.78±0.8	64.90±0.5	22.58±0.4	73.80±0.6	67.23±1.0	85.41±0.3
DE+Uniform	30.82±0.8	67.60±0.4	34.35±1.4	76.20±0.3	70.74±0.5	86.78±0.1
DE+Entropy	29.13±0.9	67.48±0.3	31.74±1.4	75.98±0.4	75.71±0.3	87.87±0.1
DE+Confidence	29.90±0.8	67.45±0.3	35.12±1.8	76.63±0.2	75.52±0.2	87.84±0.1
DE+Margin	31.82±1.3	68.85±0.4	33.42±1.3	76.18±0.2	75.49±0.8	87.89±0.2
DE+Avg-KLD	32.23±0.2	68.73±0.2	33.03±1.5	76.21±0.4	75.91±0.2	87.89±0.0
DE+CLUE	30.80±0.3	67.82±0.2	33.92±3.0	76.27±0.6	69.66±0.5	86.67±0.1
DE+BADGE	30.16±1.3	68.46±0.3	32.23±3.7	76.13±0.7	73.23±0.2	87.55±0.1
ASPEST (ours)	**37.38**±0.1	**71.61**±0.2	**38.44**±0.7	**77.69**±0.1	**77.85**±0.2	**88.28**±0.1

Table 7.3: Results of comparing ASPEST to the baselines on DomainNet R→C, Amazon Review and Otto. The mean and std of each metric over three random runs are reported (mean±std). The labeling budget M is 500. All numbers are percentages. **Bold** numbers are superior results.

Baseline comparisons. We compare ASPEST with the two existing selective classification methods: SR and DE with various active learning sampling approaches. The results in Table 7.3 (complete results on all datasets for all metrics and different labeling budgets are provided in Appendix E.3.2) show that ASPEST consistently outperforms the baselines across different image, text and tabular datasets. For example, for MNIST→SVHN, ASPEST improves the AUC from 79.36% to 88.84% when the labeling budget (M) is only 100. When $M = 500$, for CIFAR-10→CINIC-10, ASPEST improves the AUC from 90.74% to 90.95%; for FMoW, ASPEST improves the AUC from 70.59% to 71.12%; for Amazon Review, ASPEST improves the AUC from 76.63% to 77.69%; for DomainNet R→C, ASPEST improves the AUC from

68.85% to 71.61%; for DomainNet R→P, ASPEST improves the AUC from 56.67% to 58.74%; for DomainNet R→S, ASPEST improves the AUC from 46.38% to 49.62%; for Otto, ASPEST improves the AUC from 87.89% to 88.28%.

7.5.3 Analyses and Discussions

In this section, we analyze why the key components checkpoint ensembles and self-training in ASPEST can improve selective prediction and perform ablation study to show their effect.

Checkpoint ensembles can alleviate overfitting and overconfidence. We observe that in active selective prediction, when fine-tuning the model on the small amount of selected labeled test data, the model can suffer overfitting and overconfidence issues and ensembling the checkpoints in the training path can effectively alleviate these issues (see the analysis in Appendix E.3.4).

Self-training can alleviate overconfidence. We observe that the checkpoint ensemble constructed after fine-tuning is less confident on the test data U_X compared to the deep ensemble. Thus, using the softmax outputs of the checkpoint ensemble as soft pseudo-labels for self-training can alleviate overconfidence and improve selective prediction performance (see the analysis in Appendix E.3.5).

Ablation studies. Compared to DE+Margin, ASPEST has two additional components: checkpoint ensemble and self-training. We perform ablation experiments on MNIST→SVHN and DomainNet to analyze the effect of these. We also study the effect of the threshold η in self-training. The results in Table 7.4 show that for MNIST→SVHN, adding the checkpoint ensemble component alone (ASPEST without self-training) does not improve the performance over DE+Margin, whereas adding the self-training component alone (ASPEST without checkpoint ensemble) can significantly improve the performance. For DomainNet, both checkpoint ensemble and self-training have positive contributions. For both cases, ASPEST

Dataset	MNIST→SVHN		DomainNet R→C	
Metric	AUC ↑		AUC ↑	
Labeling Budget	100	500	500	1000
DE+Margin	78.59±1.4	94.31±0.6	68.85±0.4	71.29±0.3
ASPEST without self-training	78.09±1.3	94.25±0.4	69.59±0.2	72.45±0.1
ASPEST without checkpoint ensemble	83.78±2.9	96.54±0.2	69.94±0.1	72.20±0.4
ASPEST (η=0.1)	83.77±1.7	96.01±0.4	70.35±0.2	72.89±0.4
ASPEST (η=0.5)	83.99±1.3	96.24±0.2	70.92±0.3	73.37±0.1
ASPEST (η=0.6)	85.17±1.3	96.24±0.2	70.96±0.2	73.05±0.1
ASPEST (η=0.8)	85.40±2.3	96.74±0.1	71.05±0.2	72.99±0.3
ASPEST (η=0.9)	**88.84**±1.0	96.62±0.2	**71.61**±0.2	73.27±0.2
ASPEST (η=0.95)	87.67±1.3	**96.74**±0.1	71.03±0.3	**73.38**±0.2

Table 7.4: Ablation study results for ASPEST. The mean and std of each metric over three random runs are reported (mean±std). All numbers are percentages. **Bold** numbers are superior results.

(with both self-training and checkpoint ensemble) achieves much better results than DE+Margin or applying those components alone. We also show that the performance is not highly sensitive to η, while typically setting larger η (e.g. $\eta = 0.9$) yields better results.

Integrating with UDA. To study whether incorporating unsupervised domain adaption (UDA) techniques into training could improve active selective prediction, we evaluate DE with UDA and ASPEST with UDA in Appendix E.3.6. Our results show that ASPEST outperforms (or on par with) DE with UDA, although ASPEST doesn't utilize UDA. Furthermore, we show that by combining ASPEST with UDA, it might achieve even better performance. For example, on MNIST→SVHN, `ASPEST with DANN` improves the mean AUC from 96.62% to 97.03% when the labeling budget is 500. However, in some cases, combining ASPEST with UDA yields much worse results. For example, on MNIST→SVHN, when the labeling budget is 100, combining ASPEST with UDA will reduce the mean AUC by over 4%. We leave the exploration of UDA techniques to improve active selective prediction to future work – superior and robust UDA techniques can be easily incorporated into ASPEST to enhance its overall performance.

7.6 Conclusion

In this chapter, we introduced a new learning paradigm called *active selective prediction* which used active learning to improve selective prediction under distribution shift. We showed that this new paradigm resulted in improved accuracy and coverage on a distributionally shifted test domain and reduced the need for human labeling. We also proposed a novel method ASPEST using checkpoint ensemble and self-training with a low labeling cost. We demonstrated ASPEST's effectiveness over other baselines for this new problem setup on various image, text and structured datasets. Future work in this direction could investigate unsupervised hyperparameter tuning on test data, online data streaming, or further minimizing the labeling effort by designing time-preserving labeling interfaces.

8 THE TRADE-OFF BETWEEN UNIVERSALITY AND LABEL EFFICIENCY OF REPRESENTATIONS FROM CONTRASTIVE LEARNING

Contribution statement. This chapter is joint work with Zhenmei Shi, Kunyang Li, Jayaram Raghuram, Xi Wu, Yingyu Liang, and Somesh Jha. The author Jiefeng Chen contributed to part of the experiments. The paper version of this chapter appeared in ICLR 2023 (Shi et al., 2022a).

8.1 Introduction

Representation pre-training is a recent successful approach that utilizes large-scale unlabeled data to address the challenges of scarcity of labeled data and distribution shift. Different from the traditional supervised learning approach using a large labeled dataset, representation learning first pre-trains a representation function using large-scale diverse unlabeled datasets by self-supervised learning (e.g., contrastive learning), and then learns predictors on the representation using small labeled datasets for downstream target tasks. The pre-trained model is commonly referred to as a *foundation model* (Bommasani et al., 2021), and has achieved remarkable performance in many applications, e.g., BERT (Devlin et al., 2019), GPT-3 (Brown et al., 2020), CLIP (Radford et al., 2021), and Flamingo (Alayrac et al., 2022). To this end, we note that there are two properties that are key to their success: (1) *label efficiency*: with the pre-trained representation, only a small amount of labeled data is needed to learn accurate predictors for downstream target tasks; (2) *universality*: the pre-trained representation can be used across various downstream tasks.

In this chapter, we focus on *contrastive learning with linear probing* that learns a linear predictor on the representation pre-trained by contrastive learning, which is an exemplary pre-training approach (e.g., Arora et al. (2019); Chen et al. (2020a)). We highlight and study a fundamental trade-off between label efficiency and universality, though ideally, one would like to have these two key properties simultaneously.

Since pre-training with large-scale diverse unlabeled data is widely used in practice, such a trade-off merits deeper investigation.

Theoretically, we provide an analysis of the features learned by contrastive learning, and how the learned features determine the downstream prediction performance and lead to the trade-off. We propose a *hidden representation data model*, which first generates a hidden representation containing various features, and then uses it to generate the label and the input. We first show that contrastive learning is essentially generalized nonlinear PCA that can learn *hidden features invariant to the transformations* used to generate positive pairs. We also point out that additional assumptions on the data and representations are needed to obtain non-vacuous guarantees for prediction performance. We thus consider a setting where the data are generated by linear functions of the hidden representation, and formally prove that the difference in the learned features leads to the trade-off. In particular, pre-training on more diverse data learns more diverse features and is thus useful for prediction on more tasks. But it also down-weights task-specific features, implying larger sample complexity for predictors and thus worse prediction performance on a specific task. This analysis inspires us to propose a general method – *contrastive regularization* – that adds a contrastive loss to the training of predictors to improve the accuracy on downstream tasks.

Empirically, we first perform controlled experiments to reveal the trade-off. Specifically, we first pre-train on a specific dataset similar to that of the target task, and then incrementally add more datasets into pre-training. In the end, the pre-training data includes both datasets similar to the target task and those not so similar, which mimics the practical scenario that foundation models are pre-trained on diverse data to be widely applicable for various downstream tasks. Fig. 8.1 gives an example of this experiment: As we increase task diversity for contrastive learning, it increases the average accuracy on *all tasks* from 18.3% to 20.1%, while it harms the label efficiency of an individual task, on CIFAR-10 the accuracy drops from 88.5% to 76.4%. We also perform experiments on contrastive regularization, and demonstrate that it can *consistently improve* over the typical fine-tuning method across multiple datasets. In several cases, the improvement is significant: 1.3% test

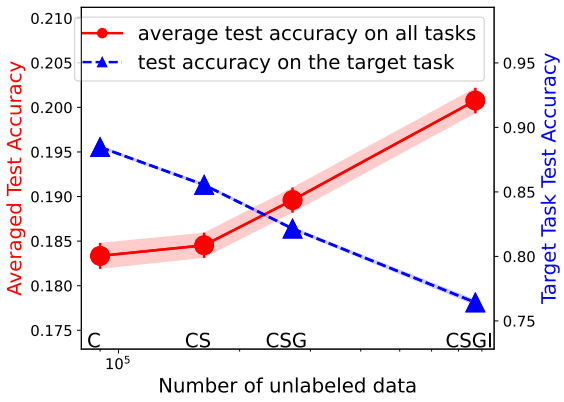


Figure 8.1: Illustration of the trade-off between universality and label efficiency. x-axis: from left to right, incrementally add CINIC-10 (C), SVHN (S), GTSRB (G), and ImageNet32 (I) for pre-training MoCo v2. For example, "CS" means CINIC-10+SVHN. The average test accuracy of prediction on all 4 datasets (red line) increases with more diverse pre-training data, while that on the target task CIFAR-10 (blue line) decreases. (The variance of the blue line is too small to be seen.) Refer to Section 8.4.1 for details.

accuracy improvement for CLIP on ImageNet, 4.8% for MoCo v3 on GTSRB (see Table 8.1 and 8.2 for details). With these results, we believe that it is of importance to bring the community's attention to this trade-off and the forward path of foundation models.

Our main contributions are summarized as follows:

- We propose a hidden representation data model and prove that contrastive learning is essentially generalized nonlinear PCA, and can encode hidden features invariant to the transformations used in positive pairs (Section 8.3.1).
- We formally prove the trade-off in a simplified setting with linear data (Section 8.3.2).
- We empirically demonstrate the trade-off across different methods and different datasets for contrastive learning with linear probing (Section 8.4.1 and 8.4.2).

- We propose a contrastive regularization method for training the predictor on a target task (Section 8.3.2), which achieves consistent improvement in our experiments (Section 8.4.3).

8.2 Related Work

Related Work on Representation Pre-training. This paradigm pre-trains a representation function on a large dataset and then uses it for prediction on various downstream tasks (Devlin et al., 2019; Kolesnikov et al., 2020; Brown et al., 2020; Newell and Deng, 2020). The representations are also called foundation models (Bommasani et al., 2021). There are mainly two kinds of approaches: (1) supervised approaches (e.g., (Kolesnikov et al., 2020)) that pre-train on large labeled datasets; (2) self-supervised approaches (e.g., (Newell and Deng, 2020)) that pre-train on large and diverse unlabeled datasets. Recent self-supervised pre-training can compete with or outperform supervised pre-training on the downstream prediction performance (Ericsson et al., 2021). Practical examples like BERT (Devlin et al., 2019), GPT-3 (Brown et al., 2020), CLIP (Radford et al., 2021), DALL·E (Ramesh et al., 2022), PaLM (Chowdhery et al., 2022) and Flamingo (Alayrac et al., 2022) have obtained effective representations universally useful for a wide range of downstream tasks.

A popular method is contrastive learning, i.e., to distinguish matching and non-matching pairs of augmented inputs (e.g., van den Oord et al. (2018); Chen et al. (2020a); He et al. (2020a); Grill et al. (2020); Chen and He (2021); Zbontar et al. (2021); Gao et al. (2021)). Some others solve "pretext tasks" like predicting masked parts of the inputs (e.g.,Doersch et al. (2015); Devlin et al. (2019)).

Related Work on Analysis of Self-supervised Pre-training. There exist abundant studies analyzing self-supervised pre-training (Arora et al., 2019; Tsai et al., 2020; Yang et al., 2020; Wang and Isola, 2020; Garg and Liang, 2020; Zimmermann et al., 2021; Tosh et al., 2021; HaoChen et al., 2021; Wen and Li, 2021; Liu et al., 2021a; Kotar et al., 2021; Van Gansbeke et al., 2021; Lee et al., 2021; Saunshi et al., 2022; Shen et al.,

2022; Kalibhat et al., 2022). They typically focus on pre-training or assume the same data distribution in pre-training and prediction. Since different distributions are the critical reason for the trade-off we focus on, we provide a new analysis. Some studies have connected contrastive learning to component analysis (Balestriero and LeCun, 2022; Tian, 2022; Ko et al., 2022). Our analysis focuses on the trade-off, while also showing a connection to PCA based on our notion of invariant features and is thus fundamentally different. Recently, Cole et al. have attempted to identify successful conditions for contrastive learning and pointed out that diverse pre-training data can decrease prediction performance compared to pre-training on the specific task data. However, they do not consider universality and provide no systematic study. Similarly, Bommasani et al. call for more research on specialization vs. diversity in pre-training data but provide no study. We aim to provide a better understanding of the trade-off between universality and label efficiency.

8.3 Theoretical Analysis

Our experiments in Section 8.4.1 demonstrate a trade-off between the universality and label efficiency of contrastively pre-trained representations when used for prediction on a distribution different from the pre-training data distribution. See Fig. 8.1 for an example. Intuitively, from the unlabeled data, pre-training can learn semantic features useful for prediction on even different data distributions. To analyze this, we need to formalize the notion of useful semantic features. So we introduce a *hidden representation data model* where a hidden representation (i.e., a set of semantic features) is sampled and then used for generating the data. Similar models have been used in some studies (HaoChen et al., 2021; Zimmermann et al., 2021), while we introduce the notion of spurious and invariant features and obtain a novel analysis for contrastive learning.

Using this theoretical model of data, Section 8.3.1 investigates what features are learned by contrastive learning. We show that *contrastive learning can be viewed as a generalization of Principal Components Analysis, and it encodes the invariant features not affected by the transformations but removes the others*. We also show that further

assumptions on the data and the representations are needed necessary for any non-vacuous bounds for downstream prediction. So Section 8.3.2 considers a simplified setting with linear data. We show that when pre-trained on diverse datasets (modeled as a mixture of unlabeled data from different tasks), it encodes all invariant features from the different tasks and thus is useful for all tasks. On the other hand, it essentially emphasizes those that are shared among the tasks, but down-weights those that are specific to a single task. Compared to pre-training only on unlabeled data from the target task, this then leads to a larger sample complexity and thus worse generalization for prediction on the target task. Therefore, we show that the trade-off between universality and label efficiency occurs due to the fact that *when many useful features from diverse data are packed into the representation, those for a specific target task can be down-weighted and thus worsen the prediction performance on it*. Based on this insight, we propose a contrastive regularization method for using representations in downstream prediction tasks, which achieves consistent improvement over the typical fine-tuning method in our experiments in Section 8.4.3.

Contrastive Learning. Let $\mathcal{X} \subseteq \mathbb{R}^d$ denote the input space, $\mathcal{Y}$ the label space, and $\overline{\mathcal{Z}} \subseteq \mathbb{R}^k$ the output vector space of the learned representation function. Let Φ denote the hypothesis class of representations $\phi : \mathcal{X} \to \overline{\mathcal{Z}}$, and $\mathcal{F}_\phi$ the hypothesis class of predictors on ϕ. A task is simply a data distribution over $\mathcal{X} \times \mathcal{Y}$. In pre-training, using transformations on unlabeled data from the tasks, we have some pre-train distribution $\mathcal{D}_{pre}$ over positive pairs (x, x^+) and negative examples x^-, where x, x^+ are obtained by applying random transformations on the same input (e.g., cropping or color jitter for images), and x^- is an independent example. The contrastive loss is $\ell\left(\phi(x)^\top(\phi(x^+) - \phi(x^-))\right)$ where $\ell(t)$ is a suitable loss function. Typically, the logistic loss $\ell(t) = \log(1 + \exp(-t))$ is used, while our analysis also holds for other loss functions. A representation ϕ is learned by:

$$\min_{\phi \in \Phi} \mathbb{E}_{(x, x^+, x^-) \sim \mathcal{D}_{pre}} \left[\ell\left(\phi(x)^\top(\phi(x^+) - \phi(x^-))\right)\right]. \tag{8.1}$$

(We simply consider the population loss since pre-training data are large-scale.) Then a predictor f is learned on top of ϕ using m labeled points $\{(x_i, y_i)\}_{i=1}^m$ from a specific target task $\mathcal{D}$:

$$\min_{f \in \mathcal{F}_\phi} \frac{1}{m} \sum_{i=1}^{m} \ell_c(f(\phi(x_i)), y_i) \tag{8.2}$$

where ℓ_c is a prediction loss (e.g. cross-entropy). Usually, f is a linear classifier (Linear Probing) with a bounded norm: $\mathcal{F}_\phi = \{f(z) = u^\top z : u \in \mathbb{R}^k, \|u\| \leqslant B\}$, where $\|\cdot\|$ denotes the ℓ_2 norm.

Hidden Representation Data Model. We now consider the pre-train distribution $\mathcal{D}_{pre}$ over (x, x^+, x^-). To capture that pre-training can learn useful features, we assume a hidden representation for generating the data: first sample a hidden representation $z \in \mathcal{Z}$ from a distribution $\mathcal{D}_z$ over some hidden representation space $\mathcal{Z} \subseteq \mathbb{R}^d$, and then generate the input x and the label y from z. (The space $\mathcal{Z}$ models semantic features, and can be different from the learned representation space $\overline{\mathcal{Z}}$.) The dimensions of z are partitioned into two disjoint subsets of $[d] := \{1, \cdots, d\}$: *spurious features* U that are affected by the transformations, and *invariant features* R that are not. Specifically, let $\mathcal{D}_U, \mathcal{D}_R$ denote the distributions of z_U and z_R, respectively, and let $x = g(z)$ denote the generative function for x. Then the positive pairs (x, x^+) are generated as follows:

$$z = [z_R; z_U] \sim \mathcal{D}_z, z_U^+ \sim \mathcal{D}_U,\ z^+ = [z_R; z_U^+], \quad x = g(z), x^+ = g(z^+). \tag{8.3}$$

That is, x, x^+ are from the same z_R but two random copies of z_U that model the random transformations. Finally, x^- is an i.i.d. sample from the same distribution as x: $z^- \sim \mathcal{D}_z, x^- = g(z^-)$.

8.3.1 What Features are Learned by Contrastive Learning?

Before analyzing prediction performance, we first analyze what features are learned in pre-training.

Contrastive Learning is Generalized Nonlinear PCA. Recall that given data x from a distribution $\mathcal{D}$, Principal Components Analysis (PCA) (Pearson, 1901; Hotelling, 1933) aims to find a linear projection function ϕ on some subspace such that the variance of the projected data $\phi(x)$ is maximized, i.e., it is minimizing the following PCA objective:

$$-\mathbb{E}_{x\sim\mathcal{D}}[\|\phi(x) - \mathbb{E}_{x'\sim\mathcal{D}}[\phi(x')]\|^2] = -\mathbb{E}_{x\sim\mathcal{D}}[\|\phi(x) - \phi_0\|^2] \tag{8.4}$$

where $\phi_0 := \mathbb{E}[\phi(x')]$ is the mean of the projected data. Nonlinear PCA replaces linear representation functions ϕ with nonlinear ones. We next show that contrastive learning is a generalization of nonlinear PCA on the smoothed representation after smoothing out the transformations.

Theorem 8.1. *If $\ell(t) = -t$, then the contrastive loss is equivalent to the PCA objective on ϕ_{z_R}:*

$$\mathbb{E}\left[\ell\left(\phi(x)^\top[\phi(x^+) - \phi(x^-)]\right)\right] = -\mathbb{E}\left[\|\phi_{z_R} - \phi_0\|^2\right] \tag{8.5}$$

where $\phi_{z_R} := \mathbb{E}[\phi(x) \mid z_R] = \mathbb{E}[\phi(g(z)) \mid z_R]$. If additionally $\phi(x)$ is linear in x, then it is equivalent to the linear PCA objective $-\mathbb{E}\left[\|\phi(\bar{x}) - \phi_0\|^2\right]$ on data $\bar{x} := \mathbb{E}[x|z_R] = \mathbb{E}[g(z)|z_R]$.

So contrastive learning is essentially nonlinear PCA when $\ell(t) = -t$, and further specializes to linear PCA when the representation is linear. As PCA finds directions with large variances, the analogue is that contrastive learning encodes important invariant features but not spurious ones.

Contrastive Learning Encodes Invariant Features and Removes Spurious Features. For a formal statement we need some weak assumptions on the data, the representations, and the loss:

(A1) z_R can be recovered from x, i.e., the inputs $x = g(z)$ from different z_R's are disjoint.

(A2) The representation functions are the *regular* functions with $\|\phi(x)\| = B_r$ $(\forall x)$ for some $B_r > 0$. Being regular means there are a finite L and a partition of $\mathcal{Z}$ into a finite number of subsets, such that in each subset all $\phi \circ g$ have Lipschitz constants bounded by L.

(A3) The loss $\ell(t)$ is convex, decreasing, and lower-bounded.

The first condition means the invariant features z_R can be extracted from x (note that g need not be invertible). The regular condition on the representation is to exclude some pathological cases like the Dirichlet function; essentially reasonable functions relevant for practice satisfy this condition, e.g., when g is Lipschitz and ϕ are neural networks with the ReLU activation. Also, note that the logistic loss typically used in practice satisfies the last condition.

We say a function $f(z)$ is independent of a subset of input dimensions z_S, if there exists a function f' such that $f(z) = f'(z_{-S})$ with probability 1, where z_{-S} denotes the set of all z_j with $j \notin S$. We say the representation ϕ encodes a feature z_i, if $\phi \circ g : \mathcal{Z} \to \overline{\mathcal{Z}}$ is not independent of z_i as long as the generative function $g(z)$ is not independent of z_i.

Theorem 8.2. *Under Assumptions* ***(A1)(A2)(A3)****, the optimal representation* ϕ^* *satisfies:*

(1) ϕ^* *does not encode the spurious features* z_U*:* $\phi^* \circ g(z)$ *is independent of* z_U*.*

(2) For any invariant feature $i \in R$*, there exists* $B_i > 0$ *such that as long as the representations' norm* $B_r \geqslant B_i$*, then* ϕ^* *encodes* z_i*. Furthermore, if* $\mathcal{Z}$ *is finite, then* B_i *is monotonically decreasing in* $\Pr[z_{R\setminus\{i\}} = z^-_{R\setminus\{i\}}, z_i \neq z^-_i]$*, the probability that in* z_R *and* z^-_R*, the* i*-th feature varies while the others remain the same.*

So contrastive learning aims to remove the spurious features and preserve the invariant features. Then the transformations should be chosen such that they will not affect the useful semantic features, but change those irrelevant to the label. Interestingly, the theorem further suggests that contrastive learning tends to favor the more "spread-out" invariant features z_i, as measured by $\Pr[z_{R\setminus\{i\}} = z^-_{R\setminus\{i\}}, z_i \neq z_i^-]$. As we increase the representation capacity B_r, B_r passes the threshold B_i for more features z_i, so ϕ^* first encodes the more spread-out invariant features and then the others.

This further suggests the following intuition for the trade-off. When pre-trained on diverse data modeled as a mixture from multiple tasks with different invariant features, the representation encodes all the invariant features and thus is useful for prediction on all the tasks. When pre-trained on only a specific task, features specific to this task are favored over those that only show up in other tasks, which leads to smaller sample complexity for learning the predictor and thus better prediction. However, to formalize this, some inductive bias assumptions about the data and the representation are necessary to get any non-vacuous guarantee for the prediction (see discussion in Appendix F.1.1). Therefore, Section 8.3.2 introduces additional assumptions and formalizes the trade-off.

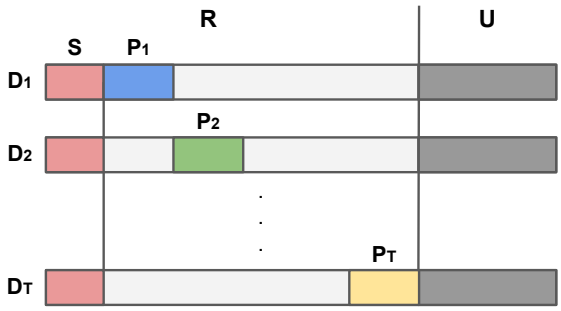


Figure 8.2: Illustration of the features in our data distributions.

8.3.2 Analyzing the Trade-Off: Linear Data

To analyze the prediction performance, we first need to model the relation between the pre-training data and the target task. We model the diverse pre-training data as a mixture of data from T different tasks $\mathcal{D}_t$'s, while the target task is one of the tasks. All tasks share a public feature set S of size s, and each task $\mathcal{D}_t$ additionally owns a private disjoint feature set P_t of size $r-s$, i.e., $P_t \cap S = \emptyset$ and $P_{t_1} \cap P_{t_2} = \emptyset$ for $t_1 \neq t_2$ (Fig. 8.2). The invariant features for $\mathcal{D}_t$ are then $R_t = S \cup P_t$. All invariant features are $R = \cup_{t=1}^{T} R_t$, and spurious features are $U = [d] \setminus R$. In task $\mathcal{D}_t$, the (x, x^+) are generated as follows:

$$z_{R_t} \sim \mathcal{N}(0, I),\ z_{R \setminus R_t} = 0,\ z_U \sim \mathcal{N}(0, I), z = [z_R; z_U], \quad x = g(z), \tag{8.6}$$

$$z_U^+ \sim \mathcal{N}(0, I), z^+ = [z_R; z_U^+], \quad x^+ = g(z^+), \tag{8.7}$$

and x^- is simply an i.i.d. copy from the same distribution as x. In practice, multiple independent negative examples are used, and thus we consider the following contrastive loss $\min_{\phi \in \Phi} \mathbb{E}_{(x,x^+)} \left[\ell \left(\phi(x)^\top (\phi(x^+) - \mathbb{E}_{x^-} \phi(x^-)) \right) \right]$ for a convex and decreasing $\ell(t)$ to pre-train a representation ϕ. Then, when using ϕ for prediction in the target task $\mathcal{D}_t$, the predictor class should contain a predictor matching the ground-truth label:

$$\mathcal{F}_{\phi,t} = \{ f(z) = u^\top z : u \in \mathbb{R}^k, \|u\| \leqslant B_{\phi,t} \} \tag{8.8}$$

where $B_{\phi,t}$ is the minimum value such that there exists $u_t \in \mathcal{F}_{\phi,t}$ with $y = u_t^\top \phi(x)$ on $\mathcal{D}_t$.

Now, given the necessity of inductive biases for non-vacuous guarantees (see Appendix F.1.1), and inspired by classic dictionary learning and recent analysis on such data (e.g., Olshausen and Field (1997); Wen and Li (2021); Shi et al. (2022b)), we assume linear data and linear representations:

- x is linear in z: $x = g(z) = Mz$ where $M \in \mathbb{R}^{d \times d}$ is an orthonormal dictionary. Since linear probing has strong performance on pre-trained representations,

we thus assume that the label in each task t is linear in its invariant features $y = (u_t^*)^\top z_{R_t}$ for some $u_t^* \in \mathbb{R}^r$.

- The representations are linear functions with weights of bounded spectral and Frobenius norms:

$$\Phi = \{\phi(x) = Wx : W \in \mathbb{R}^{k \times d}, \|W\| \leqslant 1, \|W\|_F \leqslant \sqrt{r}\}.$$

Here the norm bounds are chosen to be the minimum values to allow recovering the invariant features in the target task, i.e., there exists $\phi \in \Phi$ such that $\phi(x) = [z_{R_t}; \mathbf{0}]$.

We compare two representations: a specific one pre-trained on unlabeled data from the target task $\mathcal{D}_t$, and a universal one pre-trained on an even mixture of data from T tasks. (Appendix F.2 provides analysis for more general cases like uneven mixtures.) This captures the situation that the pre-training data contains some data similar to the target task and also other less similar data. Let $\nu_{t,1} = \sum_{j \in S} (u_t^*)_j^2$ and $\nu_{t,2} = \sum_{j \in P_t} (u_t^*)_j^2$ be the weights on the shared and task-specific invariant features, respectively. Also, assume the prediction loss ℓ_c is L-Lipschitz.

Proposition 6. *The representation ϕ^* obtained on an even mixture of data from all the tasks $\{\mathcal{D}_t : 1 \leqslant t \leqslant T\}$ satisfies $\phi^* \circ g(z) = Q\left(\sum_{j \in S} \sqrt{\alpha} z_j e_j + \sum_{j \in R \setminus S} \sqrt{\beta} z_j e_j\right)$ for some $\alpha \in [0, 1]$, $\beta = \min\left(1, \frac{r - \alpha s}{T(r-s)}\right)$, where e_j's are the basis vectors and Q is any orthonormal matrix.*
The Empirical Risk Minimizer $\hat{u} \in \mathcal{F}_{\phi^*, t}$ on ϕ^* using m labeled data points from $\mathcal{D}_t$ has risk

$$\begin{aligned}
&\mathbb{E}_{(x,y) \sim \mathcal{D}_t}[\ell_c(\hat{u}^\top \phi^*(x), y)] \\
\leqslant &4L \sqrt{\frac{1}{m}\left(\frac{\nu_{t,1}}{\alpha} + \frac{\nu_{t,2}}{\beta}\right)} \left(\sqrt{s\alpha + (r-s)\beta} + O\left(\sqrt{\frac{r}{s\alpha + (r-s)\beta}}\right)\right) \\
&+ 8\sqrt{\frac{2\ln(4/\delta)}{m}}.
\end{aligned}$$

Proposition 7. *The representation ϕ_t^* obtained on data from $\mathcal{D}_t$ satisfies $\phi_t^* \circ g(z) = Q\left(\sum_{j\in R_t} z_j e_j\right)$ where e_j's are the basis vectors and Q is any orthonormal matrix.*
The Empirical Risk Minimizer $\hat{u} \in \mathcal{F}_{\phi_t^*,t}$ on ϕ_t^* using m labeled data points from $\mathcal{D}_t$ has risk

$$\mathbb{E}_{(x,y)\sim\mathcal{D}_t}[\ell_c(\hat{u}^\top \phi_t^*(x), y)] \leqslant 4L\sqrt{\frac{r}{m}}\|u_t^*\| + 8\sqrt{\frac{2\ln(4/\delta)}{m}}.$$

While on task $\mathcal{D}_i (i \neq t)$, any linear predictor on ϕ_t^* has error at least

$$\min_u \mathbb{E}_{\mathcal{D}_i}[\ell_c(u^\top z_S, y)].$$

Difference in Learned Features Leads to the Trade-off. The key of the analysis (in Appendix F.2) is about what features are learned in the representations. Pre-trained on all T tasks, ϕ^* is a rotation of the weighted features, where the shared features are weighted by $\sqrt{\alpha}$ and task-specific ones are weighted by $\sqrt{\beta}$. Pre-trained on one task $\mathcal{D}_t$, ϕ_t^* is a rotation of the task-specific features R_t. So compared to ϕ_t^*, ϕ^* encodes all invariant features but down-weights the task-specific features P_t.

The difference in the learned features then determines the prediction performance and results in a trade-off between universality and label efficiency: compared to ϕ_t^*, ϕ^* is useful for more tasks but has worse performance on the specific task $\mathcal{D}_t$. For illustration, suppose $r = 2s$, and the shared and task-specific features are equally important for the labels on the target task: $\nu_{t,1} = \nu_{t,2} = \|u_t^*\|^2/2$. In Appendix F.2.3 we show that ϕ^* has $\alpha = 1, \beta = \frac{1}{T}$ and the error is $O\left(L\sqrt{\frac{Tr}{m}}\|u_t^*\|\right)$, while the error using ϕ_t^* is $O\left(L\sqrt{\frac{r}{m}}\|u_t^*\|\right)$. Therefore, the error when using representations pre-trained on data from T tasks is $O(\sqrt{T})$ worse than that when just pre-training on data from the target task. On the other hand, the former can be used in all T tasks and the prediction error diminishes with the labeled data number m. While the latter only encodes R_t and the only useful features on the other tasks are z_S, then even with infinite labeled data the error can be large ($\geqslant \min_u \mathbb{E}[\ell_c(u^\top z_S, y)]$, the approximation error using only the common features z_S for prediction).

Improving the Trade-off via Contrastive Regularization. The above analysis provides some guidance on improving the trade-off, in particular, improving the target prediction accuracy when given a pre-trained representation ϕ^*. It suggests that when ϕ^* is pre-trained on diverse data, one can update it by contrastive learning on some unlabeled data from the target task, which can get better features and better predictions. This is indeed the case for the illustrative example above. We can show that updating ϕ^* by contrastive learning on $\mathcal{D}_t$ can increase the weights β on the task-specific features z_{P_t}, and thus improve the generalization error (formal analysis in Appendix F.2.4).

In practice, typically one will learn the classifier and also fine-tune the representation with a labeled dataset $\{(x_i, y_i)\}_{i=1}^m$ from the target task. We thus propose *contrastive regularization* for fine-tuning: for each data point (x, y), generate contrastive pairs $\mathcal{R} = \{(\tilde{x}, \tilde{x}^+, \tilde{x}^-)\}$ by applying transformations, and add the contrastive loss on these pairs as a regularization term to the classification loss:

$$\ell_c(f(\phi(x)), y) \; + \; \frac{\lambda}{|\mathcal{R}|} \sum_{(\tilde{x}, \tilde{x}^+, \tilde{x}^-) \in \mathcal{R}} \ell\left(\phi(\tilde{x})^\top (\phi(\tilde{x}^+) - \phi(\tilde{x}^-))\right). \tag{8.9}$$

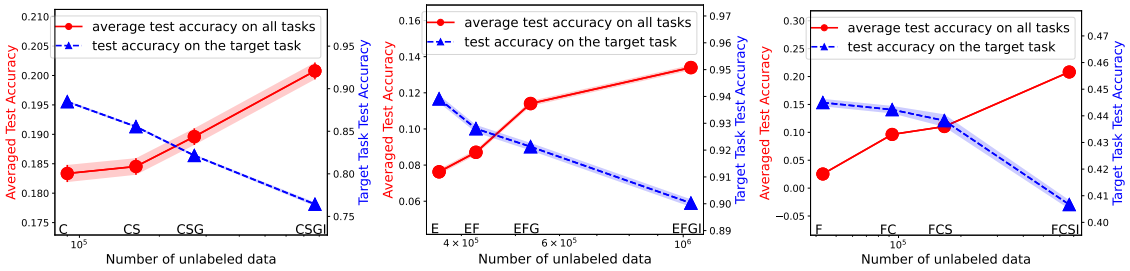
This method is simple and generally applicable to different models and algorithms. Similar ideas have been used in graph learning (Ma et al., 2021), domain generalization (Kim et al., 2021) and semi-supervised learning (Lee et al., 2022), while we use it in fine-tuning for learning predictors. Our experiments in Section 8.4.3 show that it can consistently improve the prediction performance compared to the typical fine-tuning approach.

8.4 Experiments

We conduct experiments to answer the following questions. (**Q1**) Does the trade-off between universality and label efficiency exist when training on real datasets? (**Q2**) What factors lead to the trade-off? (**Q3**) How can we alleviate the trade-off, particularly in large foundation models? Our experiments provide the following

answers: (**A1**) The trade-off widely exists in different models and datasets when pre-training on large-scale unlabeled data and adapting with small labeled data (see Section 8.4.1). This justifies our study and aligns with our analysis. (**A2**) Different datasets own many private invariant features leading to the trade-off, e.g., FaceScrub and CIFAR-10 do not share many invariant features (see Section 8.4.2). It supports our analysis in Section 8.3.2. (**A3**) Our proposed method, Finetune with Contrastive Regularization, can improve the trade-off consistently (see Section 8.4.3).

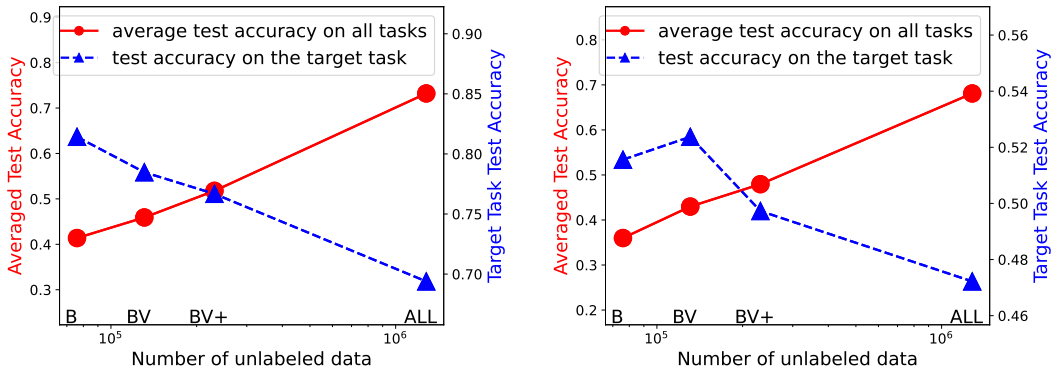
8.4.1 Verifying the Existence of the Trade-off



(a) CIFAR-10 (b) MNIST (c) Fer2013

Figure 8.3: Trade-off between universality and label efficiency for MoCo v2. x-axis: incrementally add datasets for pre-training MoCo v2. (**a**) Pre-training data: CINIC-10 (C), SVHN (S), GTSRB (G), and ImageNet32 (I). E.g., "CS" on the x-axis means CINIC-10+SVHN. Target task: CIFAR-10. Red line: average test accuracy of Linear Probing on all 4 datasets. Blue line: test accuracy on the target task. (**b**) EMNIST-Digits&Letters (E), Fashion-MNIST (F), GTSRB (G), ImageNet32 (I). Target: MNIST. (**c**) FaceScrub (F), CIFAR-10 (C), SVHN (S), ImageNet32 (I). Target: Fer2013. Note that training does *not* follow the online learning fashion, e.g., the model will pre-train from scratch (random initialization) on the CSG datasets, rather than using the model pre-trained on the CS datasets.

Evaluation & Methods. We first pre-train a ResNet18 backbone (He et al., 2016a) with different contrastive learning methods and then do Linear Probing (LP, i.e., train a linear classifier on the feature extractor) with the labeled data from the target task. We report the test accuracy on a specific target task and the average test



(a) MoCo v3 (backbone ViT-S) (b) SimSiam (backbone ResNet50)

Figure 8.4: Trade-off between universality and label efficiency on ImageNet. x-axis: from left to right, incrementally add ImageNet-Bird (B), ImageNet-Vehicle (V), ImageNet-Cat/Ball/Shop/Clothing/Fruit (+), and ImageNet (ALL) for pre-training (a) MoCo v3 with backbone ViT-S (b) SimSiam with backbone ResNet50. For example, "BV" means ImageNet-Bird + ImageNet-Vehicle. Target: ImageNet-Bird.

accuracy on all pre-training datasets (i.e., using them as the downstream tasks). Fig. 8.3 shows the results for the method MoCo v2. The size and diversity of pre-training data are increased on the x-axis by incrementally adding unlabeled training data from: (a) CINIC-10, SVHN, GTSRB, ImageNet32 (using only a 500k subset); (b) EMNIST-Digits&Letters, Fashion-MNIST, GTSRB, ImageNet32; (c) FaceScrub, CIFAR-10, SVHN, ImageNet32. We further perform larger-scale experiments on ImageNet (see Fig. 8.4).

Results. The results show that when the pre-training data becomes more diverse, the average test accuracy on all pre-training datasets increases (i.e., universality improves), while the test accuracy on the specific target task decreases (i.e., label efficiency drops). This shows a clear trade-off between universality and label efficiency. It supports our claim that diverse pre-training data allow learning diverse features for better universality, but can down-weight the features for a specific task resulting in worse prediction. This validates our theoretical analysis of the trade-off.

8.4.2 Inspecting the Trade-off: Feature Similarity

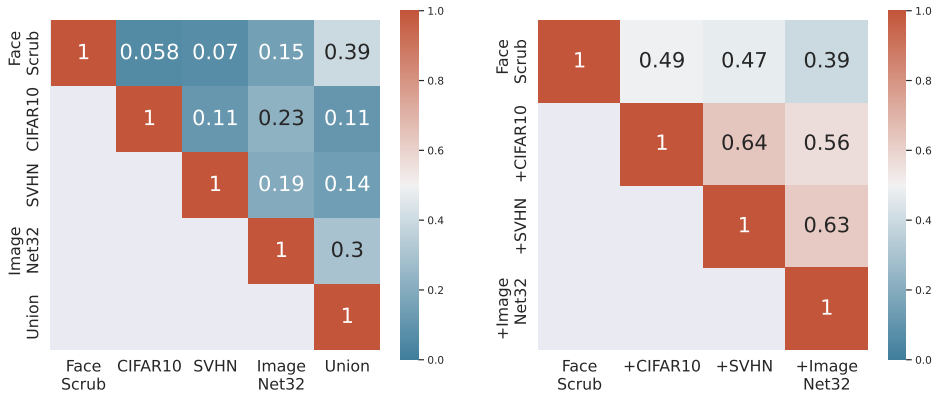
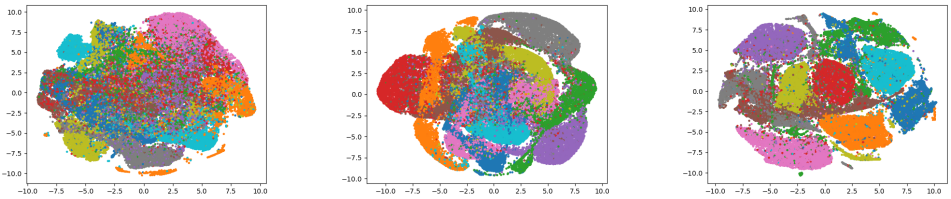


Figure 8.5: Linear CKA similarity among Fer2013 features from MoCo v2 pre-trained on different datasets. Left: each representation in the first four columns/rows is pre-trained on a single dataset. “Union" indicates the model pre-trained on the union of the four disjoint datasets. Right: from left column to right, from top row to bottom, we incrementally add datasets for pre-training.

Here we compute the similarity of the features learned from different pre-training datasets for a target task. For each pre-trained model, we extract a set of features for the target task Fer2013 using the pre-trained representation function. Then we compute the similarities between the extracted features based on different pre-training dataset pairs using linear Centered Kernel Alignment (CKA) (Kornblith et al., 2019), a widely used tool for high-dimensional feature comparison. Figure 8.5 reports the results (rows/columns are pre-training data; numbers/colors show the similarity). The left figure shows that the features from different pre-training datasets have low similarities. This is consistent with our setup in Section 8.3.2 that different tasks only share some features and each owns many private ones. The right figure shows a decreasing trend of similarity along each row. This indicates that when gradually adding more diverse pre-training data, the learned representation will encode more downstream-task-irrelevant features, and become less similar to that prior to adding more pre-training data.



(a) Linear Probing (b) Finetune (c) Ours

Figure 8.6: The t-SNE visualization (Van der Maaten and Hinton, 2008) for CIFAR-10 training data normalized features from different evaluation methods, where the model is pre-trained on (CSGI) defined in Fig. 8.3. FT and Ours are trained on the 20% CIFAR-10 training dataset. Different colors correspond to different classes.

8.4.3 Improving the Trade-off: Finetune with Contrastive Regularization

Method	**Pre-training** dataset			
	CINIC-10	+SVHN	+GTSRB	+ImageNet32
LP	88.41±0.01	85.18±0.01	82.07±0.01	75.64±0.03
FT	93.58±0.14	93.35±0.10	93.42±0.13	92.92±0.06
Ours	**94.51**±0.02	**94.26**±0.01	**94.32**±0.13	**93.66**±0.12

Table 8.1: Test accuracy on CIFAR-10 with different evaluation methods on MoCo v2 by using all CIFAR-10 training data. From left to right: incrementally add datasets for pre-training.

Evaluation & Methods. We pre-train ResNet18 by MoCo v2 as in Section 8.4.1 and report the test accuracy on CIFAR-10 when the predictor is learned by: Linear Probing (LP), Finetune (FT), and Finetune with Contrastive Regularization (Ours). LP follows the training protocol in Section 8.4.1. FT and Ours learn a linear predictor and update the representation, and use the same data augmentation for a fair comparison. FT follows MAE (He et al., 2022), while Ours uses MoCo v2 contrastive loss and regularization coefficient $\lambda = 0.1$.

Results. Table 8.1 shows that our method can consistently outperform the other baselines. In particular, it outperforms the typical fine-tuning method by about 0.7% – 1%, even when the latter also uses the same amount of data augmentation. This confirms the benefit of contrastive regularization. To further support our claim, Fig. 8.6 visualizes the features of different methods by t-SNE, showing that contrastive regularization can highlight the task-specific features and provide cleaner clustering, and thus improve the generalization, as discussed in our theoretical analysis.

Method	CLIP			MoCo v3			SimCSE	
	ImageNet	SVHN	GTSRB	CIFAR-10	SVHN	GTSRB	IMDB	AGNews
LP	77.84±0.02	63.44±0.01	86.56±0.01	95.82±0.01	61.92±0.01	75.37±0.01	86.49±0.16	87.76±0.66
FT	83.65±0.01	78.22±0.18	90.74±0.06	96.17±0.12	65.36±0.33	76.45±0.29	92.31±0.26	93.57±0.23
Ours	**84.94**±0.09	**78.72**±0.37	**92.01**±0.28	**96.71**±0.10	**66.29**±0.20	**81.28**±0.10	**92.85**±0.03	**93.94**±0.02

Table 8.2: Test accuracy for different evaluation methods on different datasets using all training data and using foundation models from CLIP, MoCo v3, and SimCSE. Data augmentation is not used for LP (Linear Probing). For FT (Finetune) and Ours (our method), 10 augmentations to each training images are used for CLIP, MoCo v3, and unique augmentation in each training step is used for SimCSE.

Larger Foundation Models. We further evaluate our method on several popular real-world large representation models (foundation models). On some of these models, the user may be able to fine-tune the representation when learning predictors. On very large foundation models, the user typically extracts feature embeddings of their data from the models and then trains a small predictor, called adapter (Hu et al., 2021; Sung et al., 2022), on these embeddings. We evaluate CLIP (ViT-L (Dosovitskiy et al., 2020) as the representation backbone), MoCo v3 (ViT-B backbone), and SimCSE (Gao et al., 2021) (BERT backbone). They are trained on (image, text), (image, image), and (text, text) pairs, respectively, so cover a good spectrum of methods. For CLIP and MoCo v3, the backbone is fixed. LP uses a linear classifier, while FT and Ours insert a two-layer ReLU network as an adapter between the backbone and the linear classification layer. Ours uses the SimCLR contrastive loss on the output of the adapter. For SimCSE, all methods use

linear classifiers. LP fixes the backbone, while FT and Ours train the classifier and fine-tune the backbone simultaneously. Ours uses the SimCSE contrastive loss on the backbone feature. We set the regularization coefficient $\lambda = 1.0$.

Table 8.2 again shows that our method can consistently improve the downstream prediction performance for all three models by about 0.4% – 4.8%, and quite significantly in some cases (e.g., 1.3% for CLIP on ImageNet, 4.8% for MoCo v3 on GTSRB). This shows that our method is also useful for large foundation models, even when the foundation models cannot be fine-tuned and only the extracted embeddings can be adapted.

8.5 Conclusion and Future Work

In this chapter, we showed and analyzed the trade-off between universality and label efficiency of representations in contrastive learning. Guided by our analysis, we proposed a contrastive regularization method to improve the trade-off. We validated our analysis and method empirically with systematic experiments using real-world datasets and foundation models. Our analysis and experiments provided insights on the conditions for this trade-off and how to effectively select the pre-training tasks.

There are many interesting open questions for future work: (1) What features does the model learn from specific pre-training and diverse pre-training datasets beyond linear data? (2) Do the other self-supervised learning methods have a similar trade-off? (3) Can we address the trade-off better to gain both properties at the same time?

9 ADAPTATION WITH SELF-EVALUATION TO IMPROVE SELECTIVE PREDICTION IN LLMS

Contribution statement. This chapter is joint work with Jinsung Yoon, Sayna Ebrahimi, Sercan O Arik, Tomas Pfister, and Somesh Jha. The author Jiefeng Chen proposed the method and completed all the experiments.

9.1 Introduction

Large Language Models (LLMs) have recently demonstrated impressive capabilities in many natural language understanding, reasoning and generation tasks, such as question answering (Jiang et al., 2021b; Singhal et al., 2023), summarization (Tang et al., 2023; Zhang et al., 2023b), semantic classification, and code generation (Poesia et al., 2022; Zhang et al., 2023a). As LLMs improve their remarkable performance, they are being increasingly considered to replace humans to perform high-stakes tasks. For example, LLMs can be used for medical QA to assist patients (Singhal et al., 2022). However, LLMs are not guaranteed to be accurate for all queries, so it is important to understand which queries they are reliable for. This information can be used to direct human oversight to the queries with the lowest selection score. *Selective prediction* (Geifman and El-Yaniv, 2017), broadly refers to the deployment scenario for AI models where humans are involved to maintain overall accuracy by reviewing AI-generated, low-confidence outputs. In this scenario, both human and AI performance are considered together to minimize human involvement cost. LLMs should be used in the real world with enhanced selective prediction performance. They should be able to assess the accuracy of their predictions and refrain from making wrong predictions. If an LLM detects that an answer might be wrong for a question, it should be able to generate an answer with the sentiment of "I don't know!" (as shown in Fig. 9.1) or defer the answer to a human for manual inspection. This will help to ensure that LLMs are used in a responsible and trustworthy manner; especially in the high-stakes contexts.

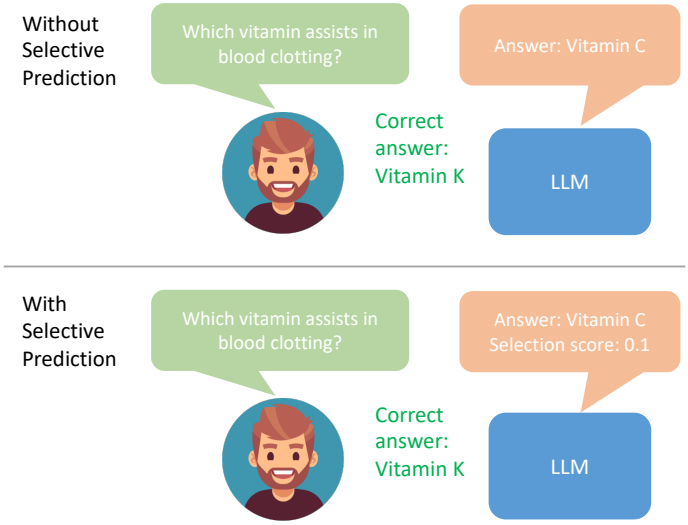


Figure 9.1: A safety-critical question from the TriviaQA dataset: "Which vitamin helps regulate blood clotting?" The OPT-2.7B model incorrectly answers "Vitamin C", when the correct answer is "Vitamin K". Without selective prediction, LLMs will directly output the wrong answer which in this case could lead users to take the wrong medicine, and thus causing potential harm. With selective prediction, LLMs will output a low selection score along with the wrong answer and can further output "I don't know!" to warn users not to trust it or verify it using other sources.

Selective prediction for LLMs is challenging because LLMs are just trained to predict the next token given a context but are not guaranteed to always predict the correct next token. Also, since LLMs generate an output sequence in an auto-regressive way, they don't directly produce a confidence score for the output sequence. Thus, obtaining selection scores from LLMs for their output sequences is not straightforward. Although there is research on selective prediction for LLMs, these studies have their own shortcomings. Kadavath et al. propose to use heuristic prompts (e.g., adding prompts like "Is the proposed answer True or False?") to trigger self-evaluation of LLMs. However, those prompts may only work for the LLM used in Kadavath et al. (2022) and may not generalize to other types of LLMs (e.g., OPT and GPT2 models evaluated in our work). Some approaches proposed using semantic entropy (Kuhn et al., 2023) or self-consistency (Wang et al., 2022) as a measure of uncertainty for selection score. However, they usually require

generating multiple output sequences to obtain the uncertainty measure for an input sequence, which introduces high computational cost and latency at test time. Fine-tuning LLMs on training data from the target question answering task using the standard LLM training loss can improve selective prediction performance. This is because fine-tuning can improve the accuracy of the predictions and maximize the likelihood of the ground-truth answer for a given question. However, maximizing the likelihood of the ground-truth answer is not the same as minimizing the likelihood of the wrong answers, since LLMs generate output sequences in an auto-regressive way. Even after fine-tuning, some wrong answers may still have high likelihood and be generated by the LLM at test time. Therefore, distinguishing correct and incorrect answers based on likelihood scores alone is a challenging task.

To address these challenges of self-evaluation and uncertainty estimation, we propose a novel framework – *Adaptation with Self-Evaluation to Improve Selective Prediction in LLMs* (*ASPIRE*). Unlike previous methods that rely on hand-crafted heuristics or multiple output sequences, our framework learns to self-evaluate from target-task data. We do this by training LLMs on a subset of the training data from the question-answering tasks. This allows the LLMs to learn to distinguish between correct and incorrect answers on their own. We then define a selection score that combines the likelihood of the generated answer with the learned self-eval score (see Eq. (9.10)) to make selective predictions. This makes our method much less computationally expensive than solutions that require generating multiple output sequences to obtain the uncertainty measure.

We conducted extensive experiments to evaluate our proposed framework, ASPIRE. We show that ASPIRE achieves state-of-the-art selective prediction performance on three question-answering datasets: CoQA, TriviaQA and SQuAD using OPT and GPT-2 models. We also provide empirical analysis to delve deeper into our proposed technique.

9.2 Related Work

Selective Prediction for LLMs. Recently, LLMs (e.g., GPT-4 (OpenAI, 2023) and PaLM (Chowdhery et al., 2022)) have achieved great success in solving various kinds of Natural Language Generation (NLG) tasks. However, LLMs are still not very reliable and may generate wrong outputs when solving NLG tasks. Due to this, selective prediction (or sometimes called selective generation (Ren et al., 2022)) is critical for safely deploying LLMs in the real-world. Different from selective prediction for classification tasks (e.g., Natural Language Inference (NLI) tasks) (Xin et al., 2021), selective prediction for LLMs in solving NLG tasks is fundamentally different since the prediction is done auto-regressively over many steps and the possible answer set has an infinite size. Recently, several work propose some uncertainty measures for LLMs, which can be used for selective prediction (Si et al., 2022; Kadavath et al., 2022; Varshney et al., 2022; Ren et al., 2022; Kuhn et al., 2023). Some recent work studies selective prediction for solving question answering tasks where questions are ambiguous (Cole et al., 2023; Yin et al., 2023). Different from previous work, our work proposes to improve the selective prediction performance of LLMs in solving question answering tasks by learning self-evaluation during fine-tuning.

Parameter Efficient Fine-tuning. Fine-tuning pretrained LLMs on downstream datasets can bring huge performance gains when compared to using the pretrained LLMs out-of-the-box (e.g., k-shot inference). However, as LLMs get larger and larger, full fine-tuning becomes very expensive in terms of computational cost and memory requirements. In addition, massive models might not be data efficient and overfitting issues might be observed, yielding suboptimal generalization. To address these issues, Parameter-Efficient Fine-tuning (PEFT) approaches have been proposed. PEFT approaches only fine-tune a small number of (extra) model parameters while freezing most parameters of the pretrained LLMs, thereby greatly decreasing the computational and storage costs. It has also been shown that PEFT approaches are better than fine-tuning in the low-data regimes and generalize better

to out-of-domain scenarios. Existing PEFT approaches include LoRA (Hu et al., 2021), Prefix Tuning (Liu et al., 2021b), Soft Prompt Tuning (Lester et al., 2021) and P-Tuning (Liu et al., 2021c). In this chapter, we propose using Soft Prompt Tuning to learn self-evaluation to improve the selective prediction performance of LLMs.

9.3 Problem Setup

Suppose we have a pre-trained LLM f for an arbitrary generative modeling task such as question answering. The output can be represented as a sequence of tokens from the vocabulary $\mathcal{V}$. Let $\mathcal{V}^*$ be the space of sequences of tokens. Suppose the logits of f on $v \in \mathcal{V}$ given $\mathbf{x} \in \mathcal{V}^*$ is $\bar{f}(v \mid \mathbf{x})$. The likelihood of the next token following $\mathbf{x}$ being v is defined as:

$$f(v \mid \mathbf{x}) := \frac{\exp\left(\bar{f}(v \mid \mathbf{x})\right)}{\sum_{v' \in \mathcal{V}} \exp\left(\bar{f}(v' \mid \mathbf{x})\right)}, \tag{9.1}$$

whereas the likelihood of generating $\hat{\mathbf{y}} \in \mathcal{V}^*$ given $\mathbf{x}$ is defined as:

$$f(\hat{\mathbf{y}} \mid \mathbf{x}) := \Pi_{i=1}^{|\hat{\mathbf{y}}|} f(\hat{y}_i \mid \mathbf{x}, \hat{\mathbf{y}}_{[i-1]}), \tag{9.2}$$

where $\hat{\mathbf{y}} = (\hat{y}_1, \ldots, \hat{y}_{|\hat{\mathbf{y}}|})$, $|\hat{\mathbf{y}}|$ is the length of $\hat{\mathbf{y}}$, $\hat{\mathbf{y}}_{[i-1]} = (\hat{y}_1, \ldots, \hat{y}_{i-1})$ for $i > 0$ and $\hat{\mathbf{y}}_{[0]} = \emptyset$. This likelihood can be very small when $|\hat{\mathbf{y}}|$ is very large. To address this issue, we define the normalized likelihood as:

$$f_{\text{norm}}(\hat{\mathbf{y}} \mid \mathbf{x}) := f(\hat{\mathbf{y}} \mid \mathbf{x})^{\frac{1}{|\hat{\mathbf{y}}|}} \tag{9.3}$$

We use f to generate the output sequence for the given input $\mathbf{x}$ by solving the following objective:

$$\hat{\mathbf{y}}^* = \arg\max_{\hat{\mathbf{y}}} \log f(\hat{\mathbf{y}} \mid \mathbf{x}) \tag{9.4}$$

It is impossible to solve this objective exactly since the output sequences can be arbitrarily long. However, we can employ some decoding strategy like greedy decoding or beam search to solve it.

To evaluate if the generated output $\hat{\mathbf{y}}$ is correct or not, we need a set of reference outputs S and an evaluation metric $M : \mathcal{V}^* \times \mathcal{V}^* \to [0, 1]$ that can evaluate the similarity of the generated output $\hat{\mathbf{y}}$ compared to the reference output $\mathbf{y}_r \in S$. With a threshold γ, we can determine the correctness of the generated output – if $\max_{\mathbf{y}_r \in S} M(\hat{\mathbf{y}}, \mathbf{y}_r) > \gamma$, then the generated output is correct; otherwise, the generated output is wrong. We discuss the specific choices of M and γ in Section 9.6.

In selective prediction, we need a rejection option, which is denoted by $\perp$. Given a training dataset $\mathcal{D}^{tr} = \{(\mathbf{x}^i, \mathbf{y}^i)\}_{i=1}^{n_{tr}}$ randomly sampled from a target task distribution, we aim to build a selective predictor $f_s : \mathcal{V}^* \to \mathcal{V}^* \cup \{\perp\}$ that can achieve strong selective prediction performance on the test dataset $\mathcal{D}^{te} = \{(\mathbf{x}^i, S^i)\}_{i=1}^{n_{te}}$, where S^i is the set of reference outputs for the input $\mathbf{x}^i$. The selective predictor f_s is composed of a predictor $\hat{f} : \mathcal{V}^* \to \mathcal{V}^*$ and a selection scoring function $g : \mathcal{V}^* \to \mathbb{R}$. With $\hat{f}$ and g, the selective predictor f_s is proposed as:

$$f_s(\mathbf{x}; \tau) = \begin{cases} \hat{f}(\mathbf{x}) & \text{if } g(\mathbf{x}) \geqslant \tau, \\ \perp & \text{if } g(\mathbf{x}) < \tau \end{cases}, \tag{9.5}$$

where τ is a threshold. The accuracy of the selective predictor is defined as the fraction of the accepted inputs where the predictions are correct. The coverage of the selective predictor is defined as the fraction of the inputs that are accepted. We can tune the threshold τ to achieve a certain coverage and there would be an accuracy-coverage trade-off.

We use the area under the accuracy-coverage curve (AUACC) metric to measure selective prediction performance and use the area under the receiver operator characteristic curve (AUROC) metric to measure the quality of the selection score estimation. AUROC is equivalent to the probability that a randomly chosen correct output sequence has a higher selection score than a randomly chosen incorrect output sequence.

9.4 ASPIRE Framework

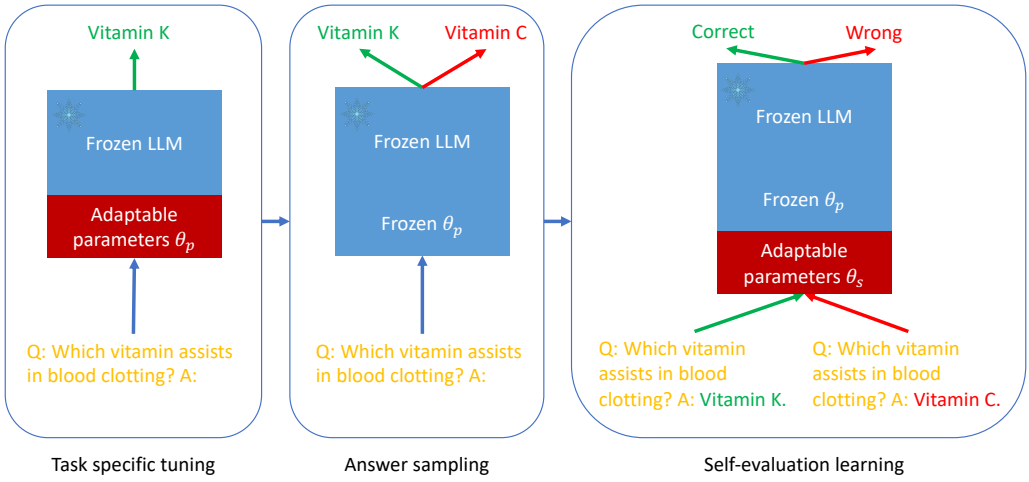


Figure 9.2: In the proposed framework ASPIRE, we first perform task specific tuning to train adaptable parameters θ_p while freezing the LLM. Then we use the LLM with the learned θ_p to generate different answers for each training question to create a dataset for self-evaluation learning. Finally, we train the adaptable parameters θ_s to learn self-evaluation using the created dataset while freezing the LLM and the learned θ_p.

We propose that LLMs should have the self-evaluation ability such that they should be able to distinguish whether their proposed answers for a given question are correct or not. Although some previous work (Kadavath et al., 2022) show that LLMs have good self-evaluation ability with specially designed prompts, those prompts may not transfer to different kinds of LLMs (as shown by our experiments and in Kuhn et al. (2023)) and hand-crafting prompts for different kinds of LLMs can be expensive. A more effective approach is to collect some training data to employ self-evaluation. Towards this end, we propose a novel framework – Adaptation with Self-Evaluation to Improve Selective Prediction in LLMs (ASPIRE). Fig. 9.2 illustrates the proposed framework and the details are explained next.

Given a training dataset for a generative task, we can fine-tune the pre-trained LLM on the training data to improve its prediction performance. Towards this end, parameter efficient tuning techniques (e.g., soft prompt tuning (Lester et al., 2021) and LoRA (Hu et al., 2021)) might be employed to adapt the pre-trained

LLM on the task, given their effectiveness in obtaining strong generalization with small amount of target task data. Specifically, the model parameters θ of the LLM are frozen and adaptable parameters θ_p are added for fine-tuning. Only θ_p are updated to solve the following training objective:

$$\min_{\theta_p} \mathbb{E}_{(\mathbf{x},\mathbf{y})\sim\mathcal{D}^{tr}} \mathcal{L}(\mathbf{x}, \mathbf{y}; \theta, \theta_p), \tag{9.6}$$

where $\mathcal{L}$ is the LLM training loss (e.g. cross-entropy). Such fine-tuning can improve selective prediction performance because it not only improves the prediction accuracy, but also enhances the likelihood of correct output sequences.

To further improve selective prediction performance, we propose to fine-tune the LLM to learn self-evaluation. We first use the LLM with the learned θ_p to generate different answers for each example $(\mathbf{x}, \mathbf{y}) \in \mathcal{D}^{tr}$. Suppose the decoding algorithm used to generate output sequences for each input $\mathbf{x}$ is $\mathcal{A}$. $\mathcal{A}$ would produce a list of generated output sequences $\mathcal{A}(f, \theta_p, \mathbf{x}) = [\hat{\mathbf{y}}^1, \ldots, \hat{\mathbf{y}}^k]$, where k is the number of output sequences generated. We aim to generate output sequences that have high likelihood (i.e., $f(\hat{\mathbf{y}}^j \mid \mathbf{x}; \theta_p)$ is high). We use the metric M defined in Section 9.3 to determine if the generated output $\hat{\mathbf{y}}^j$ is correct or not. If $M(\hat{\mathbf{y}}^j, \mathbf{y}) > \hat{\gamma}$, we label $\hat{\mathbf{y}}^j$ as a correct output for $\mathbf{x}$; otherwise, we label $\hat{\mathbf{y}}^j$ as a wrong output for $\mathbf{x}$. Here, the threshold $\hat{\gamma}$ might be different from the threshold γ used for evaluation. We choose a sufficiently large value of $\hat{\gamma}$ (e.g., $\hat{\gamma} = 0.9$) so that the generated wrong outputs wouldn't be labeled as correct outputs. In Appendix G.4, we provide more details and analyses on selection of $\hat{\gamma}$.

After sampling high-likelihood outputs for each query, we add adaptable parameters θ_s and only tune θ_s for learning self-evaluation. Since the output sequence generation only depends on θ and θ_p, freezing θ and the learned θ_p can avoid changing the prediction behaviors of the LLM when learning self-evaluation. Let z_c and z_w be a pair of tokens that represent the words "correct" and "wrong" respectively.

We can then optimize θ_s using the following training objective:

$$\begin{aligned}
&\min_{\theta_s} \quad \mathbb{E}_{(\mathbf{x},\mathbf{y})\sim\mathcal{D}^{\text{tr}}} \quad \mathcal{L}_c + \mathcal{L}_w \\
&\mathcal{L}_c = \mathbb{E}_{\hat{\mathbf{y}}\sim S_c(\mathbf{x},\mathbf{y})} - \log f(z_c|\mathbf{x},\hat{\mathbf{y}};\theta_p,\theta_s) \\
&\mathcal{L}_w = \mathbb{E}_{\hat{\mathbf{y}}\sim S_w(\mathbf{x},\mathbf{y})} - \log f(z_w|\mathbf{x},\hat{\mathbf{y}};\theta_p,\theta_s)
\end{aligned} \tag{9.7}$$

where $S_c(\mathbf{x},\mathbf{y})$ is a set of correct outputs containing the reference output $\mathbf{y}$ and k_c correct outputs with highest likelihood from $\mathcal{A}(f,\theta_p,\mathbf{x})$, and $S_w(\mathbf{x},\mathbf{y})$ is a set of wrong outputs containing k_w wrong outputs with highest likelihood from $\mathcal{A}(f,\theta_p,\mathbf{x})$. If $\mathcal{A}(f,\theta_p,\mathbf{x})$ has less than k_c correct outputs (or has less than k_w wrong outputs), we include all its correct outputs (or all its wrong outputs) in S_c (or S_w). We ensure that S_w contains at least one wrong output. If $\mathcal{A}(f,\theta_p,\mathbf{x})$ doesn't contain wrong outputs, we add a default wrong output (e.g., the empty string) to S_w.

After training θ_p and θ_s, we obtain the prediction for the query $\mathbf{x}$ via solving the following objective:

$$\hat{\mathbf{y}}^* = \arg\max_{\hat{\mathbf{y}}} \log f(\hat{\mathbf{y}} \mid \mathbf{x};\theta_p). \tag{9.8}$$

We use the beam search decoding method towards this. We define the likelihood of the output $\hat{\mathbf{y}}^*$ being correct for the query $\mathbf{x}$ as:

$$P(z_c \mid \mathbf{x},\hat{\mathbf{y}}^*) = \frac{\exp\left(\bar{f}(z_c \mid \mathbf{x},\hat{\mathbf{y}}^*;\theta_p,\theta_s)\right)}{\sum_{z\in\{z_c,z_w\}}\exp\left(\bar{f}(z \mid \mathbf{x},\hat{\mathbf{y}}^*;\theta_p,\theta_s)\right)} \tag{9.9}$$

This score $P(z_c \mid \mathbf{x},\hat{\mathbf{y}}^*)$ is referred as the learned self-eval score. Overall, the selection scoring function is proposed as:

$$g(\mathbf{x}) = (1-\alpha)\cdot\log f_{\text{norm}}(\hat{\mathbf{y}}^* \mid \mathbf{x};\theta_p) + \alpha\cdot\log P(z_c \mid \mathbf{x},\hat{\mathbf{y}}^*). \tag{9.10}$$

where $\alpha \in [0,1]$ is a hyper-parameter.

9.5 Implementation via Soft Prompt Tuning

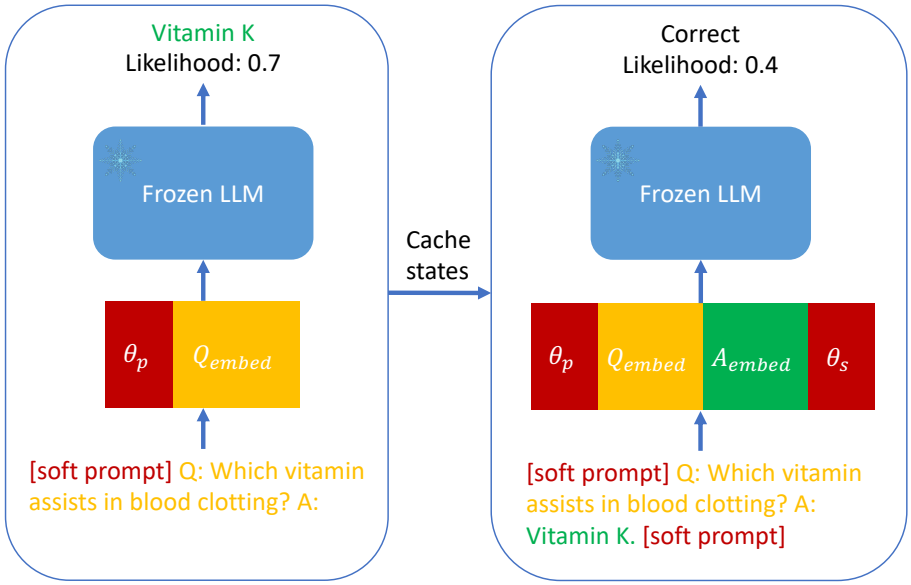


Figure 9.3: Implementation of the proposed framework via soft prompt tuning. θ_p and θ_s are learnable soft prompt embeddings. Q_{embed} and A_{embed} are input embeddings for the question and answer respectively. We first generate the answer and the likelihood of the answer, and then compute the learned self-eval score. We can cache the states when generating the answer and reuse those states when computing the learned self-eval score to save computational costs.

In the proposed framework, θ_p and θ_s can be trained using parameter efficient tuning approaches. In this chapter, we focus on Soft Prompt Tuning, as illustrated in Fig. 9.3. The driving force behind this approach lies in the recognition that if we can develop prompts that effectively stimulate self-evaluation, it should be possible to discover these prompts through soft prompt tuning in conjunction with targeted training objectives.

We first briefly introduce the soft prompt tuning method proposed by Lester et al. (2021). We consider LLMs based on the Transformer architecture (Vaswani et al., 2017). Given a query $\mathbf{x} = (x_1, \ldots, x_{m_q})$, Transformers first embed the tokens, forming a matrix $X \in \mathbb{R}^{m_q \times d_e}$, where d_e is the dimension of the embedding space. The soft-prompts are represented as parameters $\tilde{\theta} \in \mathbb{R}^{l \times d_e}$, where l is the length

of the prompt. The prompt is then concatenated to the embedded input forming a single matrix $[\tilde{\theta}; X] \in \mathbb{R}^{(m_q+l)\times d_e}$, which then flows through the transformer as normal.

In the proposed framework, we need to train two portions of the prompts $\theta_p \in \mathbb{R}^{l\times d_e}$ and $\theta_s \in \mathbb{R}^{l\times d_e}$. Utilizing soft prompt tuning, the training objective (9.6) is proposed as:

$$\min_{\theta_p} \mathbb{E}_{(\mathbf{x},\mathbf{y})\sim\mathcal{D}^{tr}} \frac{1}{|\mathbf{y}|} \sum_{j=1}^{|\mathbf{y}|} -\log f(y_j|[\theta_p; X; Y_{[j-1]}]), \tag{9.11}$$

where X is the embedding of $\mathbf{x}$ and $Y_{[j-1]}$ is the embedding of $y_{[j-1]}$. On the other hand, the training objective (9.7) is proposed as:

$$\begin{aligned} &\min_{\theta_s} \quad \mathbb{E}_{(\mathbf{x},\mathbf{y})\sim\mathcal{D}^{tr}} \quad \mathcal{L}_c + \mathcal{L}_w \\ &\mathcal{L}_c = \mathbb{E}_{\hat{\mathbf{y}}\sim S_c(\mathbf{x},\mathbf{y})} - \log f(z_c|[\theta_p; X; \hat{Y}; \theta_s]) \\ &\mathcal{L}_w = \mathbb{E}_{\hat{\mathbf{y}}\sim S_w(\mathbf{x},\mathbf{y})} - \log f(z_w|[\theta_p; X; \hat{Y}; \theta_s]) \end{aligned} \tag{9.12}$$

where $\hat{Y}$ is the embedding of $\hat{\mathbf{y}}$. The inference objective (9.8) in the framework becomes:

$$\hat{\mathbf{y}}^* = \arg\max_{\hat{\mathbf{y}}} \log f(\hat{\mathbf{y}} \mid [\theta_p; X]) \tag{9.13}$$

The learned self-eval score $P(z_c \mid \mathbf{x}, \hat{\mathbf{y}}^*)$ becomes:

$$P(z_c \mid \mathbf{x}, \hat{\mathbf{y}}^*) = \frac{\exp\left(\bar{f}(z_c \mid [\theta_p; X; \hat{Y}^*; \theta_s])\right)}{\sum_{z\in\{z_c,z_w\}} \exp\left(\bar{f}(z \mid [\theta_p; X; \hat{Y}^*; \theta_s])\right)} \tag{9.14}$$

where $\hat{Y}^*$ is the embedding of $\hat{\mathbf{y}}^*$.

To generate the output sequence and obtain the selection score for a given input sequence, we employ two stages: first, we obtain the generated output and the

likelihood for the generated output and then, we obtain the learned self-eval score. Since the query of the second stage is constructed by appending some additional tokens to the query of the first stage, the second stage can reuse the states in the first stage instead of recomputing them to save some computational cost (see Fig. 9.3).

Lastly, we note that the computational complexity of the proposed method at test time is $O(l_{max})$ with l_{max} being the maximum length of the generated output sequence. In Appendix G.2, we provide a more detailed analysis of the computational complexity of different methods. The predictive entropy and semantic entropy methods have a complexity of $O(m \cdot l_{max})$ where m is the number of output sequences sampled for uncertainty estimation, which is much larger than that of our method.

9.6 Experiments

Our experimental evaluation is focused on the following questions:

(**Q1**) Could a learning-based system using self-evaluation improve selective prediction in LLMs compared to other post-hoc selective prediction alternatives?

(**A1**) By learning self-evaluation, we can significantly improve selective prediction performance across different datasets and LLMs (see Table 9.2).

(**Q2**) What kinds of decoding algorithms could be used as $\mathcal{A}$ for the proposed framework ASPIRE?

(**A2**) Using decoding algorithms that can sample different high-likelihood answers as $\mathcal{A}$ (e.g., beam search) is important for ASPIRE to achieve good selective prediction performance (see Table 9.4).

(**Q3**) What is the effect of the number of training samples for the proposed method ASPIRE?

(**A3**) More training samples lead to enhanced performance and with ~2K samples, ASPIRE can outperform the baselines without soft prompt tuning significantly on different datasets (see Table 9.5).

9.6.1 Setup

Dataset. We focus on the free-form question answering tasks on the datasets CoQA (Reddy et al., 2019), TriviaQA (Joshi et al., 2017) and SQuAD (Rajpurkar et al., 2016). For CoQA and SQuAD, since each question is asked based on a context paragraph, we evaluate the LLMs in the zero-shot setting. For TriviaQA, since the LLMs have limited accuracy under the zero-shot setting, we evaluate the LLMs in 5-shot setting. For each dataset, we use a subset of the original training set containing 50K examples for adapting LLMs by default. The details of the datasets are given in Appendix G.1.1.

LLMs. We use OPT (Zhang et al., 2022) and GPT-2 (Radford et al., 2019) models of various sizes. For OPT, we consider OPT-350M, OPT-1.3B and OPT-2.7B. For GPT-2, we consider GPT2-Medium, GPT2-Large and GPT2-XL. The details of these models are given in Appendix G.1.2.

Baselines. For selective prediction, we need to get a predicted output sequence $\hat{\mathbf{y}}^*$ and a selection score $g(\mathbf{x})$ for each input sequence $\mathbf{x}$ given a model f. The model f can be a pre-trained LLM or an adapted LLM with θ_p trained using the training objective (9.11). We use the beam-search decoding to obtain the predicted output sequence $\hat{\mathbf{y}}^*$ and consider the following baselines to compute the selection score $g(\mathbf{x})$: (1) Perplexity; (2) Predictive Entropy; (3) Semantic Entropy (Kuhn et al., 2023); (4) Self-eval; (5) P(True) (Kadavath et al., 2022). More details can be found in Appendix G.1.3.

Evaluation metrics. We use the Rouge-L (Lin and Och, 2004) as the evaluation metric M to evaluate the similarity of the generated answer to the reference answers

following Kuhn et al. (2023). For the threshold γ that is used to determine the correctness of the generated answer, we consider relatively larger values of γ since we focus on safety-critical applications where accepting a wrong answer is more costly compared to rejecting a correct answer that is different from the reference answers (refer to Appendix G.3 for the justifications of the choices of γ). Unless specified, we use $\gamma = 0.7$ as default.

Training hyper-parameters. We have two stages of training: the first stage is to train the soft prompt θ_p using the training objective (9.11) and the second stage is to train the soft prompt θ_s using the training objective (9.12). For both stages, we train the soft prompts for 10 epochs using AdamW optimizer with a batch size of 8, a learning rate of 0.01 and cosine learning rate scheduling. More training details can be found in Appendix G.1.4.

ASPIRE setup. We use the beam search as the decoding algorithm $\mathcal{A}$. We set the number of beams equal to k and use the k highest scoring beams as the answer list $\mathcal{A}(f, \theta_p, \mathbf{x})$. We set $l = 50$, $\hat{\gamma} = 0.9$, $k = 10$, $k_c = 2$, $k_w = 10$ and $\alpha = 0.25$ by default. We choose these hyper-parameters based on the performance on the validation set from TriviaQA using the OPT-2.7B model. We then use the same hyper-parameters for all datasets and models.

9.6.2 Results

Model	CoQA	TriviaQA	SQuAD
	Acc ↑	Acc ↑	Acc ↑
Pre-trained GPT2-XL	46.27	11.80	7.41
Adapted GPT2-XL with θ_p	69.18	17.45	75.44
Pre-trained OPT-2.7B	60.68	21.38	35.95
Adapted OPT-2.7B with θ_p	80.45	29.21	83.27

Table 9.1: Results of evaluating the accuracy of different LLMs. All numbers are percentages.

Model	Method	CoQA		TriviaQA		SQuAD	
		AUACC ↑	AUROC ↑	AUACC ↑	AUROC ↑	AUACC ↑	AUROC ↑
Pre-trained GPT2-XL	Perplexity	55.93	62.05	22.60	72.88	7.68	51.90
	Predictive Entropy	60.76	67.53	24.83	76.20	10.04	57.21
	Semantic Entropy	63.03	70.50	24.37	75.33	10.38	59.17
	Self-eval	46.67	50.83	9.30	42.75	7.32	49.56
	P(True)	46.98	51.17	10.62	44.54	10.69	60.87
Adapted GPT2-XL with θ_p	Perplexity	83.27	72.79	36.49	79.92	88.73	75.08
	Predictive Entropy	83.49	73.44	37.31	82.21	88.25	74.16
	Semantic Entropy	84.40	75.16	36.68	81.40	88.62	75.26
	Self-eval	69.91	51.90	14.39	43.33	74.26	49.13
	P(True)	70.63	52.83	13.59	40.59	74.34	49.09
	ASPIRE (ours)	**85.65**	**78.32**	**38.06**	**83.23**	**89.86**	**78.35**
Pre-trained OPT-2.7B	Perplexity	75.26	70.16	40.93	78.86	40.82	57.20
	Predictive Entropy	75.29	69.16	41.20	78.92	47.18	62.85
	Semantic Entropy	76.31	70.96	40.72	78.06	51.53	68.40
	Self-eval	62.32	52.26	25.88	59.04	41.78	59.05
	P(True)	62.16	51.80	24.88	56.89	34.77	49.42
Adapted OPT-2.7B with θ_p	Perplexity	90.80	74.23	53.56	81.74	92.86	75.72
	Predictive Entropy	90.63	72.87	53.91	82.19	92.96	75.58
	Semantic Entropy	91.23	74.61	53.58	81.55	93.21	76.53
	Self-eval	81.30	50.76	32.98	56.03	86.34	56.99
	P(True)	81.14	51.01	33.48	56.27	82.59	49.48
	ASPIRE (ours)	**92.63**	**80.25**	**55.06**	**84.44**	**94.73**	**82.60**

Table 9.2: Results of evaluating different methods to compute the selection score when the model's predictions are fixed. All numbers are percentages. **Bold** numbers are superior results.

We first evaluate the accuracy of different LLMs. The results in Table 9.1 show that after training θ_p via soft prompt tuning, the accuracy of LLMs is improved significantly. We then evaluate different methods to compute the selection score when the model's predictions are fixed. The results in Table 9.2 show that the proposed method ASPIRE significantly outperforms the baselines in terms of the AUACC and AUROC metrics across different datasets and LLMs. The results also show that after soft prompt tuning, the AUACC of different methods is significantly improved due to the improvement of the accuracy and the perplexity becomes more meaningful in separating correct and wrong answers. Additional results in Appendix G.5 show that ASPIRE significantly outperforms the baselines across OPT and GPT2 models of different sizes for different values of the Rouge threshold

γ.

9.6.3 Empirical Analyses

Model	Method	CoQA		TriviaQA		SQuAD	
		AUACC ↑	AUROC ↑	AUACC ↑	AUROC ↑	AUACC ↑	AUROC ↑
Adapted OPT-2.7B with θ_p	ASPIRE ($\alpha = 0.0$)	90.80	74.23	53.56	81.74	92.86	75.72
	ASPIRE ($\alpha = 0.25$)	**92.63**	**80.25**	**55.06**	**84.44**	**94.73**	**82.60**
	ASPIRE ($\alpha = 0.5$)	92.56	80.18	54.61	84.33	94.59	82.16
	ASPIRE ($\alpha = 0.75$)	92.05	78.37	52.71	81.52	94.28	80.98
	ASPIRE ($\alpha = 1.0$)	91.33	76.08	48.84	76.39	93.77	79.48

Table 9.3: Results of studying the effect of the hyper-parameter α in the proposed selection score (Eq. (9.10)). All numbers are percentages. **Bold** numbers are superior results.

Model	Decoding Algorithm	CoQA		TriviaQA		SQuAD	
		AUACC ↑	AUROC ↑	AUACC ↑	AUROC ↑	AUACC ↑	AUROC ↑
Adapted GPT2-XL with θ_p	Multinomial (T=0.1)	83.82	74.22	36.40	80.67	89.75	77.56
	Multinomial (T=1.0)	84.96	76.15	37.03	81.41	**90.12**	**78.71**
	Multinomial (T=2.0)	83.06	72.96	36.34	80.14	89.41	76.98
	Beam search	**85.65**	**78.32**	**38.06**	**83.23**	89.86	78.35
Adapted OPT-2.7B with θ_p	Multinomial (T=0.1)	92.04	77.96	55.09	84.28	94.24	80.52
	Multinomial (T=1.0)	92.60	79.86	**55.15**	84.29	94.57	82.08
	Multinomial (T=2.0)	92.02	77.91	53.80	82.40	94.15	80.42
	Beam search	**92.63**	**80.25**	55.06	**84.44**	**94.73**	**82.60**

Table 9.4: Results of comparing different decoding algorithms for answer sampling in the proposed method. We denote the temperature as T. All numbers are percentages. **Bold** numbers are superior results.

The effect of α. We study the effect of the hyper-parameter α in the proposed selection score (Eq. (9.10)). The results in Table 9.3 show that setting $\alpha = 0.25$ leads to the best performance since it combines the normalized likelihood and the learned self-eval score in a good way. Only using the normalized likelihood (i.e., $\alpha = 0$) or only using the learned self-eval score (i.e., $\alpha = 1$) leads to much worse performance. In practice, the value of α can be chosen based on the performance on the validation data.

Model	Method	CoQA		TriviaQA		SQuAD	
		AUACC ↑	AUROC ↑	AUACC ↑	AUROC ↑	AUACC ↑	AUROC ↑
Pre-trained OPT-2.7B	Predictive Entropy	75.29	69.16	41.20	78.92	47.18	62.85
	Semantic Entropy	76.31	70.96	40.72	78.06	51.53	68.40
Adapted OPT-2.7B with θ_p	ASPIRE (size=1k)	80.87	67.01	45.70	78.98	85.42	71.42
	ASPIRE (size=2k)	85.71	73.72	46.64	79.24	88.27	75.74
	ASPIRE (size=5k)	87.83	74.58	49.77	82.06	90.09	77.09
	ASPIRE (size=10k)	90.46	78.29	51.88	83.13	92.48	79.46
	ASPIRE (size=50k)	92.63	80.25	55.06	84.44	94.73	82.60

Table 9.5: Results of studying the effect of training set size for the proposed ASPIRE. All numbers are percentages.

The choices of $\mathcal{A}$. We compare two decoding algorithms – beam search and multinomial sampling that can be used as $\mathcal{A}$ for answer sampling. For beam search, we use the k highest scoring beams as the answer list. For multinomial sampling, we consider `temperature` (denoted as T) in the set $\{0.1, 1.0, 2.0\}$. The results in Table 9.4 show that using multinomial sampling with $T = 2.0$ or $T = 0.1$ leads to worse performance compared to other decoding algorithms. If we set a high temperature ($T = 2.0$) for multinomial sampling, then we sample some random answers that might not have high-likelihood. If we set a low temperature ($T = 0.1$) for multinomial sampling, then we repeatedly sample the same high-likelihood answers. Thus, the results suggest that sampling different high-likelihood answers is important for our method to achieve high accuracy and coverage in selective prediction. The results also show that using beam search leads to similar performance as using multinomial sampling with $T = 1$. So we can use either one in practice.

Training sample efficiency. We perform experiments to study the effect of the number of training samples for ASPIRE. We fix the number of training steps to be 50K while varying the size of the training dataset. The results in Table 9.5 show that more training samples lead to performance improvement and with 2K training samples, ASPIRE can outperform the baselines without soft prompt tuning by a large margin across different datasets. This underlines that our method, ASPIRE, can significantly improve selective prediction performance even with limited number of training samples.

9.7 Conclusion

In this chapter, we proposed a novel framework for adaptation with self-evaluation to improve selective prediction in LLMs. We implemented the framework via soft prompt tuning and demonstrated its superior performance over existing methods through extensive experiments. In future work, one could explore implementing our framework via other parameter efficient tuning approaches and applying our method to larger LLMs.

10 CONCLUSION AND FUTURE WORK

In this thesis, we have undertaken a comprehensive exploration of the challenges presented by distribution shifts in deep learning and have proposed solutions across three research directions: robust deep learning, reliable model deployment, and foundation models. Our research findings and contributions have shed light on the development of deep learning systems that are more robust and reliable, capable of effectively addressing real-world scenarios.

In the first research direction, robust deep learning, our focus was on enhancing the adversarial robustness of deep neural networks (DNNs). We introduced novel defenses, such as transductive-learning based defenses and defenses based on rejection, with the aim of mitigating the impact of adversarial attacks. Through the formulation of these defenses and their evaluation using strong attack frameworks, we have provided valuable insights into achieving robust predictions. Furthermore, we extended our investigation beyond prediction and explored robustness in other aspects of deep learning, including robust attribution and robust out-of-distribution detection. Our research in this direction establishes a solid foundation for the development of DNNs that can withstand various forms of attacks and maintain their performance under distribution shifts.

The second research direction, reliable model deployment, addressed the challenge of deploying DNNs in real-world scenarios. We proposed a new framework for estimating the generalization capabilities of DNNs during test time. Additionally, we introduced the concept of active selective prediction, leveraging human input to enhance the reliability of model deployment. By involving humans in the decision-making process, we achieved better utilization of human expertise and improved the overall reliability of the deployed models. Our research in this direction contributes to ensuring the safe and trustworthy deployment of DNNs in practical applications.

In the third research direction, foundation models, we provided new insights into the training of foundation models. We discovered a trade-off between the universality and label efficiency of model representations trained through con-

trastive learning. These insights provide valuable guidance for training foundation models that can generalize well across diverse tasks and adapt to distribution shifts. Additionally, we proposed a novel framework for adaptation with self-evaluation to improve the selective prediction performance of large language models (LLMs).

The research presented in this thesis underscores the importance of addressing distribution shifts in deep learning systems. Through our investigation of robust deep learning, reliable model deployment, and foundation models, we have made significant contributions towards tackling these challenges. However, there are still areas that require further exploration and improvement.

One such area is the development of more advanced defenses against adversarial attacks. Although our proposed defenses have demonstrated promising adversarial robustness against strong adaptive attacks, there remains significant potential for further improvement. Additionally, the field of adversarial machine learning is rapidly evolving, with new attack strategies emerging. One idea for improving robustness is to leverage foundation models, which encode prior knowledge and can be used to generate data for adversarial training. Therefore, continuous research and development in this domain are necessary to ensure the ongoing effectiveness of defenses.

Moreover, the involvement of humans in the decision-making process within reliable model deployment remains an area of interest. Further investigation into effective strategies for integrating human expertise and feedback can lead to improved model performance and increased trust in deployed systems. One potential approach is to utilize human feedback for fine-tuning LLMs, aligning their behaviors with human expectations.

Lastly, ongoing research efforts are needed for the training and adaptation of foundation models. As the landscape of downstream tasks and data distributions continues to evolve, it becomes crucial to develop techniques that enable efficient adaptation and generalization of foundation models. One idea is to explore new training objectives for training and adapting foundation models, enabling them to have an awareness of their capabilities and limitations.

In conclusion, this thesis has provided valuable insights and proposed solutions

to address the challenges posed by distribution shifts in deep learning. Through our focus on robust deep learning, reliable model deployment, and foundation models, we have contributed to the development of more robust, reliable, and adaptable deep learning systems. The findings and methodologies presented here lay the groundwork for further advancements in the field, leading to more effective and trustworthy deep learning systems in real-world applications. Continued research and innovation in these areas will pave the way for future progress and advancements in the field of deep learning.

A APPENDIX FOR CHAPTER 3

A.1 Additional Theory and Proofs

A.1.1 Alternate Definition of $R_\epsilon^{\mathbf{rej}}(f, \alpha)$

Proposition 8. *For any $\epsilon \geqslant 0$ and $\alpha \in [0, 1]$, the metric $R_\epsilon^{rej}(f, \alpha)$ can be equivalently defined as*

$$R_\epsilon^{rej}(f, \alpha) \;=\; \mathop{\mathbb{E}}_{(x,y)\sim\mathcal{D}} \Big[\max_{x' \in \mathcal{N}(x,\, \alpha\epsilon)} \mathbf{1}\big\{f(x') = \perp\big\} \;\vee\; \max_{x'' \in \mathcal{N}(x,\epsilon)} \mathbf{1}\big\{f(x'') \notin \{y, \perp\}\big\} \Big]. \quad \text{(A.1)}$$

This definition allows us to interpret the metric $R_\epsilon^{rej}(f, \alpha)$ as consisting of two types of errors: 1) errors due to rejecting small perturbations within the $\alpha\epsilon$-neighborhood and 2) mis-classification errors within the ϵ-neighborhood.

Proof. Consider the original definition of $R_\epsilon^{\text{rej}}(f, \alpha)$ in Eq. (3.5)

$$R_\epsilon^{\text{rej}}(f, \alpha) \;=\; \mathop{\mathbb{E}}_{(x,y)\sim\mathcal{D}} \Big[\max_{x' \in \mathcal{N}(x,\, \alpha\epsilon)} \mathbf{1}\big\{f(x') \neq y\big\} \;\vee\; \max_{x'' \in \mathcal{N}(x,\epsilon)} \mathbf{1}\big\{f(x'') \notin \{y, \perp\}\big\} \Big]. \quad \text{(A.2)}$$

The error in the first term inside the expectation can be split into the error due to rejection and the error due to mis-classification, i.e.,

$$\mathbf{1}\big\{f(x') \neq y\big\} \;=\; \mathbf{1}\big\{f(x') = \perp\big\} \;\vee\; \mathbf{1}\big\{f(x') \notin \{y, \perp\}\big\}.$$

The maximum of this error over the $\alpha\epsilon$-neighborhood can be expressed as

$$\max_{x' \in \mathcal{N}(x,\, \alpha\epsilon)} \mathbf{1}\big\{f(x') \neq y\big\} \;=\; \max_{x' \in \mathcal{N}(x,\, \alpha\epsilon)} \mathbf{1}\big\{f(x') = \perp\big\} \;\vee\; \max_{x' \in \mathcal{N}(x,\, \alpha\epsilon)} \mathbf{1}\big\{f(x') \notin \{y, \perp\}\big\},$$

which is easily verified for binary indicator functions. Substituting the above result

into Eq. (A.2), we get

$$\begin{aligned} R_\epsilon^{\text{rej}}(f,\alpha) &= \mathop{\mathbb{E}}_{(x,y)\sim\mathcal{D}}\Big[\max_{x'\in\mathcal{N}(x,\,\alpha\epsilon)} \mathbf{1}\big\{f(x')=\perp\big\} \vee \max_{x'\in\mathcal{N}(x,\,\alpha\epsilon)} \mathbf{1}\big\{f(x')\notin\{y,\perp\}\big\} \\ &\qquad \vee \max_{x''\in\mathcal{N}(x,\epsilon)} \mathbf{1}\big\{f(x'')\notin\{y,\perp\}\big\}\Big] \\ &= \mathop{\mathbb{E}}_{(x,y)\sim\mathcal{D}}\Big[\max_{x'\in\mathcal{N}(x,\,\alpha\epsilon)} \mathbf{1}\big\{f(x')=\perp\big\} \vee \max_{x''\in\mathcal{N}(x,\epsilon)} \mathbf{1}\big\{f(x'')\notin\{y,\perp\}\big\}\Big]. \end{aligned}$$

In the last step, we combined the second and third terms inside the expectation using the observation that

$$\begin{aligned} &\max_{x'\in\mathcal{N}(x,\,\alpha\epsilon)} \mathbf{1}\big\{f(x')\notin\{y,\perp\}\big\} \vee \max_{x'\in\mathcal{N}(x,\epsilon)} \mathbf{1}\big\{f(x')\notin\{y,\perp\}\big\} \\ = &\max_{x'\in\mathcal{N}(x,\epsilon)} \mathbf{1}\big\{f(x')\notin\{y,\perp\}\big\}. \end{aligned}$$

This shows the equivalence of the two definitions of $R_\epsilon^{\text{rej}}(f,\alpha)$. □

The definition in Eq. (A.1) serves as motivation for the first adaptive attack to create adversarial examples x' within the neighborhood $\mathcal{N}(x,\alpha\epsilon)$ that are rejected by the defense method.

A.1.2 Proof of Lemma 3.3

We first prove a more general lemma in Lemma A.1, from which integration by parts gives Lemma 3.3. Note that the step rejection loss and ramp rejection loss satisfy the conditions in Lemma 3.3 which gives them a simpler expression to compute the total robust loss.

Lemma A.1. *Let* $s(\alpha) := R_\epsilon^{rej}(f,\alpha)$ *and assume it is right-semicontinuous. For any monotonically non-increasing* $\ell^{rej}:[0,\infty)\to[0,1]$*, the total robust loss can be computed by:*

$$L_\epsilon(f;\ell^{rej}) = R_\epsilon^{rej}(f,0) + \big(\ell^{rej}(0)-1\big)\,p_{rej} + \int_0^1 \ell^{rej}(\alpha\epsilon)\,ds(\alpha), \tag{A.3}$$

where the integral is the Riemann–Stieltjes integral, and

$$p_{rej} := \Pr\left[f(\mathbf{x}) = \perp \wedge \left(\forall \mathbf{x}' \in \mathcal{N}(\mathbf{x}, \epsilon),\ f(\mathbf{x}') \in \{y, \perp\}\right)\right]$$

is the probability that the clean input $(\mathbf{x}, y)$ *is rejected and no perturbations of* $\mathbf{x}$ *within the* ϵ*-ball are misclassified.*

Proof. Let $\mathcal{W}$ denote the event that there exists $\mathbf{x}' \in \mathcal{N}(\mathbf{x}, \epsilon)$ such that $\mathbf{x}'$ is misclassified, i.e., $f(\mathbf{x}') \notin \{y, \perp\}$. Let $\mathcal{C}$ denote the event that the clean input $\mathbf{x}$ is rejected, i.e., $f(\mathbf{x}) = \perp$. Clearly, $p_{\text{rej}} = \Pr[\mathcal{C} \setminus \mathcal{W}]$ where $\setminus$ is the set minus. Finally, let $\mathcal{R}$ denote the event that there exists $\mathbf{x}' \in \mathcal{N}(\mathbf{x}, \epsilon)$ such that $\mathbf{x}'$ is rejected.

We only consider $\mathcal{W} \cup \mathcal{C} \cup \mathcal{R}$, since otherwise $(\mathbf{x}, y)$ contributes a loss 0 to $L_\epsilon(f; \ell^{\text{rej}})$. The union can be partitioned into three disjoint subsets.

- $\mathcal{W}$: Such an $(\mathbf{x}, y)$ contributes a loss 1. Since $R_\epsilon^{\text{rej}}(f, 0) = \Pr[\mathcal{W} \cup \mathcal{C}]$, we have $\Pr[\mathcal{W}] = R_\epsilon^{\text{rej}}(f, 0) - \Pr[\mathcal{C} \setminus \mathcal{W}] = R_\epsilon^{\text{rej}}(f, 0) - p_{\text{rej}}$. Then this subset contributes a loss $R_\epsilon^{\text{rej}}(f, 0) - p_{\text{rej}}$.

- $\mathcal{C} \setminus \mathcal{W}$: Such a data point contributes a loss $\ell^{\text{rej}}(0)$, given the assumption that ℓ^{rej} is monotonically non-increasing. Then this subset contributes a loss $\ell^{\text{rej}}(0) p_{\text{rej}}$.

- $\mathcal{R} \setminus (\mathcal{W} \cup \mathcal{C})$: That is, there exists no $\mathbf{x}' \in \mathcal{N}(\mathbf{x}, \epsilon)$ that is misclassified, the clean input $\mathbf{x}$ is accepted, but there exists some $\mathbf{x}' \in \mathcal{N}(\mathbf{x}, \epsilon)$ that is rejected. Let L_3 denote the loss contributed by this subset.

Now, it is sufficient to show that

$$L_3 = \int_0^1 \ell^{\text{rej}}(\alpha\epsilon)\, \mathrm{d}s(\alpha).$$

For any positive integer t, let $a_0 = 0 \leqslant a_1 \leqslant \cdots \leqslant a_{t-1} \leqslant a_t = 1$ be an arbitrary

sequence on $[0,1]$. Let

$$\begin{aligned} M_i &= \sup\{\ell^{\text{rej}}(\alpha\epsilon),\ \ a_{i-1} \leqslant \alpha \leqslant a_i\}, \\ m_i &= \inf\{\ell^{\text{rej}}(\alpha\epsilon),\ \ a_{i-1} \leqslant \alpha \leqslant a_i\}, \\ U(s,\ell^{\text{rej}}) &= \sum_{i=1}^{t} M_i\,(s(a_i) - s(a_{i-1})), \\ L(s,\ell^{\text{rej}}) &= \sum_{i=1}^{t} m_i\,(s(a_i) - s(a_{i-1})). \end{aligned}$$

Let $\mathcal{R}_i$ denote the event that there exists $\mathbf{x}'$ such that $a_{i-1}\epsilon < d(\mathbf{x}',\mathbf{x}) \leqslant a_i\epsilon$ and $\mathbf{x}'$ is rejected. Then $\mathcal{R} = \cup_{i=1}^{t}\mathcal{R}_i$, and

$$\begin{aligned} s(a_i) - s(a_{i-1}) &= R_\epsilon^{\text{rej}}(f,a_i) - R_\epsilon^{\text{rej}}(f,a_{i-1}) \\ &= \Pr\left[\mathcal{R}_i \setminus (\cup_{j=1}^{i-1}\mathcal{R}_j) \setminus (\mathcal{W}\cup\mathcal{C})\right]. \end{aligned}$$

Since ℓ^{rej} is monotonically non-increasing, each data point in $\mathcal{R}_i \setminus (\cup_{j=1}^{i-1}\mathcal{R}_j) \setminus (\mathcal{W}\cup\mathcal{C})$ should contribute a loss that is within $[m_i, M_i]$. Therefore,

$$L(s,\ell^{\text{rej}}) \ \leqslant\ L_3 \ \leqslant\ U(s,\ell^{\text{rej}})$$

for any sequence $\{a_i\}_{i=0}^{t}$. When $s(\alpha)$ is right-semicontinous, the Riemann–Stieltjes integral exists, and $L_3 = \int_0^1 \ell^{\text{rej}}(\alpha\epsilon)\,ds(\alpha)$. This completes the proof. □

A.1.3 Proof of Theorem 3.4

Theorem A.2 (Restatement of Theorem 3.4)**.** *Consider binary classification. Let $f^*(\mathbf{x})$ be any classifier without a rejection option. For any $\delta \in [0,1]$ and $\epsilon \geqslant 0$, there exists a selective classifier f_δ, whose robust error curve is bounded by:*

$$R_\epsilon^{rej}(f_\delta,\alpha) \ \leqslant\ R_{\epsilon'}(f^*),\ \forall\alpha\in[0,1] \tag{A.4}$$

where $\epsilon' = \max\{(\alpha+\delta)\epsilon, (1-\delta)\epsilon\}$. *Moreover, the bound is tight: for any* $\alpha \in [0,1]$, *there exist simple data distributions and* f^* *such that any* f *must have* $R^{rej}_{\epsilon}(f,\alpha) \geqslant R_{\epsilon'}(f^*)$.

Proof. For any $r > 0$, let $\mathcal{N}(f^*, r)$ denote the region within distance r to the decision boundary of f^*:

$$\mathcal{N}(f^*, r) := \{\mathbf{x} \in \mathcal{X} : \exists \mathbf{x}' \in \mathcal{N}(\mathbf{x}, r) \text{ and } f^*(\mathbf{x}') \neq f^*(\mathbf{x})\}.$$

Consider a parameter $\delta \in [0,1]$ and construct a selective classifier f_δ as follows:

$$f_\delta(\mathbf{x}) := \begin{cases} \perp & \text{if } \mathbf{x} \in \mathcal{N}(f^*, \delta\epsilon), \\ f^*(\mathbf{x}) & \text{otherwise.} \end{cases} \tag{A.5}$$

We will show that any sample $(\mathbf{x}, y)$ contributing error to $R^{rej}_{\epsilon}(f_\delta, \alpha)$ must also contribute error to $R_{\epsilon'}(f^*)$, where $\epsilon' = \max\{(\alpha+\delta)\epsilon, (1-\delta)\epsilon\}$. This will prove that $R^{rej}_{\epsilon}(f_\delta, \alpha) \leqslant R_{\epsilon'}(f^*)$.

Consider the following two cases:

- Consider the first type of error in $R^{rej}_{\epsilon}(f_\delta, \alpha)$: $\max_{\mathbf{x}' \in \mathcal{N}(\mathbf{x}, \alpha\epsilon)} \mathbf{1}[f_\delta(\mathbf{x}') \neq y] = 1$. This implies that there exists $\mathbf{x}' \in \mathcal{N}(\mathbf{x}, \alpha\epsilon)$ such that $f_\delta(\mathbf{x}') \neq y$. So there are two subcases to consider:

  (1) $\mathbf{x}' \in \mathcal{N}(f^*, \delta\epsilon)$: in this case $\mathbf{x} \in \mathcal{N}(f^*, (\delta+\alpha)\epsilon)$.

  (2) $f^*(\mathbf{x}') \neq y$: in this case either $f^*(\mathbf{x}) \neq y$, or $f^*(\mathbf{x}) = y \neq f^*(\mathbf{x}')$ and thus $\mathbf{x} \in \mathcal{N}(f^*, \alpha\epsilon)$.

  In summary, either $f^*(\mathbf{x}) \neq y$ or $\mathbf{x} \in \mathcal{N}(f^*, (\alpha+\delta)\epsilon)$.

- Next consider the second type of error in $R^{rej}_{\epsilon}(f_\delta, \alpha)$:

$$\max_{\mathbf{x}'' \in \mathcal{N}(\mathbf{x}, \epsilon)} \mathbf{1}[f_\delta(\mathbf{x}'') \notin \{y, \perp\}] = 1.$$

  This means there exists an $\mathbf{x}'' \in \mathcal{N}(\mathbf{x}, \epsilon)$ such that $f_\delta(\mathbf{x}'') \notin \{y, \perp\}$, i.e., $\mathbf{x}'' \notin \mathcal{N}(f^*, \delta\epsilon)$ and $f^*(\mathbf{x}'') \neq y$. This implies that *all* $\mathbf{z} \in \mathcal{N}(\mathbf{x}'', \delta\epsilon)$ should have

$f^*(\mathbf{z}) = f^*(\mathbf{x}'') \neq y$. In particular, there exists $\mathbf{z} \in \mathcal{N}(\mathbf{x}'', \delta\epsilon)$ with $d(\mathbf{z}, \mathbf{x}) \leqslant (1-\delta)\epsilon$ and $f^*(\mathbf{z}) \neq y$. It can be verified that $\mathbf{z} = \delta\mathbf{x} + (1-\delta)\mathbf{x}''$, which is a point on the line joining $\mathbf{x}$ and $\mathbf{x}''$, satisfies the above condition. In summary, either $f^*(\mathbf{x}) \neq y$, or $f^*(\mathbf{x}) = y \neq f^*(\mathbf{z})$ and thus $\mathbf{x} \in \mathcal{N}(f^*, (1-\delta)\epsilon)$.

Overall, a sample $(\mathbf{x}, y)$ contributing error to $R_\epsilon^{\text{rej}}(f_\delta, \alpha)$ must satisfy either $f^*(\mathbf{x}) \neq y$ or $\mathbf{x} \in \mathcal{N}(f^*, \epsilon')$. Clearly, such a sample also contributes an error to $R_{\epsilon'}(f^*)$. Therefore, we have

$$R_\epsilon^{\text{rej}}(f_\delta, \alpha) \leqslant R_{\epsilon'}(f^*), \forall \alpha \in [0,1]. \tag{A.6}$$

To show that the bound is tight, consider the following data distribution. Let $\mathbf{x} \in \mathbb{R}$ and $y \in \{-1, +1\}$, $\alpha \in [0,1]$, and let $\beta \in (0, \frac{1}{2})$ be some constant. Let the distribution be: $(\mathbf{x}, y)$ is $(-4\epsilon, -1)$ with probability $\frac{1-\beta}{2}$, $(-\frac{\alpha\epsilon}{4}, -1)$ with probability $\frac{\beta}{2}$, $(\frac{\alpha\epsilon}{4}, +1)$ with probability $\frac{\beta}{2}$, and $(4\epsilon, +1)$ with probability $\frac{1-\beta}{2}$. Let $\delta = \frac{1-\alpha}{2}$, $f^*(\mathbf{x}) := \text{sign}(\mathbf{x} + \epsilon)$. It is clear that $R_{\epsilon'}(f^*) = R_{(1+\alpha)\epsilon/2}(f^*) = \frac{\beta}{2}$. It is also clear that any f must have $R_\epsilon^{\text{rej}}(f, \alpha) \geqslant \frac{\beta}{2}$ since the points $-\frac{\alpha\epsilon}{4}$ and $\frac{\alpha\epsilon}{4}$ have distance only $\frac{\alpha\epsilon}{2}$ but have different labels. □

We note that the proof generalizes that of Theorem 5 in Tramèr (2021). In particular, our theorem includes the latter as a special case (corresponding to $\alpha = 0$ and $\delta = \frac{1}{2}$).

A.1.4 Comparing the Robustness Curves and Total Robust Losses of f_δ and f^* in Theorem 3.4

Theorem 3.4 investigates the feasibility of a good selective classifier (w.r.t. the robustness curve and the total robust loss), by constructing a classifier f_δ. It is meaningful to perform a fine-grained analysis into when f_δ has a better total robust loss than the classifier f^* without rejection.

For simplicity, we assume f^* has a 0 standard error on the clean data distribution. The analysis for the case with a nonzero standard error is similar.

Theorem A.3. *Consider binary classification. Let $f^*(\mathbf{x})$ be any classifier without a rejection option that has 0 standard error on the data distribution. Suppose the data density for data points with a distance of r to the decision boundary of f^* is $p(r)$. Consider the selective classifier f_δ defined in Theorem 3.4:*

$$f_\delta(\mathbf{x}) := \begin{cases} \perp & \text{if } \mathbf{x} \in \mathcal{N}(f^*, \delta\epsilon), \\ f^*(\mathbf{x}) & \text{otherwise.} \end{cases} \tag{A.7}$$

Then for $\delta \in [0, \frac{1}{2}]$, the total robust loss of f_δ with respect to the rejection loss ℓ^{rej} can be computed by:

$$L_\epsilon(f_\delta; \ell^{rej}) = \int_0^{(1-\delta)\epsilon} p(r)dr + \int_{(1-\delta)\epsilon}^{(1+\delta)\epsilon} \ell^{rej}(r - \delta\epsilon)\, p(r)dr. \tag{A.8}$$

Also, the curve of robust error of f_δ can be computed by:

$$R_\epsilon^{rej}(f_\delta, \alpha) = \begin{cases} \int_0^{(1-\delta)\epsilon} p(r)dr & \text{if } \alpha \in [0, 1-2\delta], \\ \int_0^{(\alpha+\delta)\epsilon} p(r)dr & \text{if } \alpha \in (1-2\delta, 1]. \end{cases} \tag{A.9}$$

Proof. First consider the total robust loss. When $0 < r \leqslant (1-\delta)\epsilon$, the data point will contribute a loss of 1 to the total robust loss. When $(1-\delta)\epsilon < r \leqslant (1+\delta)\epsilon$, the data point will contribute a loss of $\ell^{\text{rej}}(r - \delta\epsilon)$ to the total robust loss. When $r > (1+\delta)\epsilon$, the data point will contribute 0 loss to the total robust loss. Thus, we have Eq. (A.8).

For the curve of robust error, the data points with $0 < r \leqslant (1-\delta)\epsilon$ will always contribute a loss of 1. When $\alpha \in [0, 1-2\delta]$, no data points with $r \geqslant (1-\delta)\epsilon$ can contribute a loss, either by small perturbations to get a rejection or mis-classification, or by large perturbations to get a mis-classification. When $\alpha \in (1-2\delta, 1]$, data points with $r \in [(1-\delta)\epsilon, (\alpha+\delta)\epsilon]$ will contribute a loss of 1 by small perturbations to get a rejection. Thus, we have Eq. (A.9). □

Now we are ready to compare f_δ to f^*.

First consider the curve of robust error. It is easy to know that the curve of robust error of f^* is $R_\epsilon^{\text{rej}}(f^*, \alpha) = \int_0^\epsilon p(r)dr$ for $\alpha \in [0, 1]$. When $\alpha \in [0, 1-\delta]$, we have $R_\epsilon^{\text{rej}}(f^*, \alpha) \geqslant R_\epsilon^{\text{rej}}(f_\delta, \alpha)$; when $\alpha \in (1-\delta, 1]$, we have $R_\epsilon^{\text{rej}}(f^*, \alpha) \leqslant R_\epsilon^{\text{rej}}(f_\delta, \alpha)$. When $\alpha \in [0, 1-2\delta]$, if $\int_{(1-\delta)\epsilon}^\epsilon p(r)dr$ is large, then f_δ will have much lower robust error with rejection than f^*, since $R_\epsilon^{\text{rej}}(f^*, \alpha) - R_\epsilon^{\text{rej}}(f_\delta, \alpha) = \int_{(1-\delta)\epsilon}^\epsilon p(r)dr$; when $\alpha \in (1-2\delta, 1-\delta]$, if $\int_{(\alpha+\delta)\epsilon}^\epsilon p(r)dr$ is large, then f_δ will also have much lower robust error with rejection than f^*, since $R_\epsilon^{\text{rej}}(f^*, \alpha) - R_\epsilon^{\text{rej}}(f_\delta, \alpha) = \int_{(\alpha+\delta)\epsilon}^\epsilon p(r)dr$.

Now consider the total robust loss. The total robust loss of f^* is $\int_0^\epsilon p(r)dr$, which is larger than that of f_δ by $\int_{(1-\delta)\epsilon}^\epsilon p(r)dr - \int_{(1-\delta)\epsilon}^{(1+\delta)\epsilon} \ell^{\text{rej}}(r-\delta\epsilon)\, p(r)dr$. More precisely, both f^* and f_δ will get mis-classification loss on some perturbations of points with distance in the range $[0, (1-\delta)\epsilon]$ from the decision boundary of f^*. This is because there always exist some large perturbations of these points crossing the boundary. On the other hand, f^* simply gets mis-classification loss from perturbations of the points with distance in the range $[(1-\delta)\epsilon, \epsilon]$. While for all points with distance larger than $(1-\delta)\epsilon$, f_δ can correctly classify all their small perturbations of magnitude at most $(1-2\delta)\epsilon$, and reject or correctly classify their larger perturbations. So it only gets rejection loss for large magnitudes, which can then potentially lead to smaller total robust losses than f^* for monotonically non-increasing rejection losses.

Therefore, for some data distributions, there exists a fixed δ such that f_δ can get small total robust losses with respect to a wide range of reasonable rejection losses. For example, consider the step rejection losses $\ell^{\text{rej}}(r) = \mathbf{1}\{r \leqslant \alpha_0\epsilon\}$ with parameter in the range $\alpha_0 \in [0, \bar{\alpha}_0]$ where $\bar{\alpha}_0 \in [0, 1]$. If we set $\delta = \frac{1-\bar{\alpha}_0}{2}$, then $1-2\delta = \bar{\alpha}_0$ and $\alpha \in [0, 1-2\delta]$. The total robust loss of f_δ with respect to these rejection losses is $\int_0^{\frac{(1+\bar{\alpha}_0)}{2}\epsilon} p(r)dr$. In contrast, the total robust loss of f^* is $\int_0^\epsilon p(r)dr$, which can be significantly larger than that of f_δ. The total robust loss of f_δ is smaller than that of f^* by the amount $\int_{\frac{(1+\bar{\alpha}_0)}{2}\epsilon}^\epsilon p(r)dr$. If the probability mass of points with $r \in [\frac{(1+\bar{\alpha}_0)}{2}\epsilon, \epsilon]$ is large, then the improvement is significant.

A.2 Calculating the Total Robust Loss

In this section, we discuss the calculation of the total robust loss for specific instantiations of the rejection loss ℓ^{rej}. Given the curve of robust error $\{s(\alpha) := R_\epsilon^{\text{rej}}(f, \alpha) \,:\, \alpha \in [0,1]\}$ and a specific choice of rejection loss $\ell^{\text{rej}}(r)$ that is monotonically non-increasing, we can use Eq. (A.3) from Lemma A.1 to calculate the total robust loss:

$$L_\epsilon(f; \ell^{\text{rej}}) \;=\; R_\epsilon^{\text{rej}}(f, 0) \;+\; (\ell^{\text{rej}}(0) \,-\, 1)\, p_{\text{rej}} \;+\; \int_0^1 \ell^{\text{rej}}(\alpha\epsilon)\, ds(\alpha). \tag{A.10}$$

As discussed in Corollary 3.3, let us additionally choose the rejection loss to be differentiable almost everywhere, and satisfy the conditions $\ell^{\text{rej}}(0) = 1$ and $\ell^{\text{rej}}(\epsilon) = 0$. This is satisfied e.g., by the ramp and step rejection losses defined in Eqs. (3.4) and (3.3). Applying the product rule (or integration by parts), the integral term in the total robust loss can be expressed as

$$\begin{aligned}\int_0^1 \ell^{\text{rej}}(\alpha\epsilon)\, ds(\alpha) \;&=\; \ell^{\text{rej}}(\epsilon)\, s(1) \;-\; \ell^{\text{rej}}(0)\, s(0) \;-\; \int_0^1 s(\alpha)\, d\ell^{\text{rej}}(\alpha\epsilon) \\ &=\; -\, s(0) \;-\; \int_0^1 s(\alpha)\, d\ell^{\text{rej}}(\alpha\epsilon).\end{aligned}$$

Substituting the above into Eq. (A.10), the total robust loss simplifies into

$$L_\epsilon(f; \ell^{\text{rej}}) \;=\; -\int_0^1 s(\alpha)\, d\ell^{\text{rej}}(\alpha\epsilon). \tag{A.11}$$

From the above expression, we next calculate the total robust loss for the ramp and step rejection losses.

A.2.1 Ramp Rejection loss

Recall that the ramp rejection loss is defined as

$$\ell^{\text{rej}}(r) \;=\; \left(1 - \frac{r}{\epsilon}\right)^t, \;\; r \in [0, \epsilon]$$

for some $t \geqslant 1$. We have $\ell^{\text{rej}}(\alpha\epsilon) \;=\; (1-\alpha)^t$ and

$$d\ell^{\text{rej}}(\alpha\epsilon) \;=\; -t\,(1 - \alpha)^{t-1}\,d\alpha.$$

Substituting the above into Eq. (A.11) gives the total robust loss

$$L_\epsilon(f; \ell^{\text{rej}}) \;=\; t\int_0^1 s(\alpha)\,(1 - \alpha)^{t-1}\,d\alpha.$$

For the special case $t = 1$, this reduces to the area under the robust error curve $\int_0^1 s(\alpha)\,d\alpha$.
For the special case $t = 2$, this reduces to $2\int_0^1 s(\alpha)\,(1-\alpha)\,d\alpha$ (and so on for larger t).
In our experiments, we calculate the total robust loss for $t \in \{1, 2, 3, 4\}$. Since we calculate the robust error curve only for a finite set of α values, we approximate the above integrals using the trapezoidal rule. We use finely-spaced α values in $[0, 1]$ with a spacing of 0.01, and use linear interpolation to obtain intermediate (missing) values of the robust error curve.

A.2.2 Step Rejection loss

Recall that the step rejection loss is defined as

$$\ell^{\text{rej}}(r) \;=\; \mathbf{1}\{r \;\leqslant\; \alpha_0\epsilon\}, \;\; r \in [0, \epsilon]$$

for some $\alpha_0 \in [0, 1]$. In this case, there are two ways to calculate the total robust loss, both leading to the same result.

Approach 1 For the step rejection loss, the derivative can be defined by appealing to the Dirac delta function $\delta(x)$. Specifically, $\ell^{\text{rej}}(\alpha\epsilon) = \mathbf{1}\{\alpha \leqslant \alpha_0\}$ and

$$\mathrm{d}\ell^{\text{rej}}(\alpha\epsilon) \;=\; -\,\delta(\alpha - \alpha_0)\,\mathrm{d}\alpha,$$

where the negative sign arises because the step loss drops from 1 to 0 at $\alpha = \alpha_0$. Substituting the above into Eq. (A.11) gives the total robust loss

$$L_\epsilon(f; \ell^{\text{rej}}) \;=\; \int_0^1 s(\alpha)\,\delta(\alpha - \alpha_0)\,\mathrm{d}\alpha \;=\; s(\alpha_0) \;=\; R_\epsilon^{\text{rej}}(f, \alpha_0).$$

Approach 2 We directly use the definition of the total robust loss in Eq. (A.10), which for the step rejection loss is

$$\begin{aligned} L_\epsilon(f; \ell^{\text{rej}}) \;&=\; R_\epsilon^{\text{rej}}(f, 0) \;+\; \int_0^1 \mathbf{1}\{\alpha \leqslant \alpha_0\}\,\mathrm{d}s(\alpha) \\ &=\; s(0) \;+\; \int_0^{\alpha_0} 1\,\mathrm{d}s(\alpha) \;=\; s(0) \;+\; s(\alpha_0) \;-\; s(0) \\ &=\; s(\alpha_0) \;=\; R_\epsilon^{\text{rej}}(f, \alpha_0). \end{aligned}$$

For the step rejection loss, the total robust loss is equal to the value of the robust error curve at the point $\alpha = \alpha_0$.

A.3 Designing Adaptive Attacks

To compute the robust accuracy with rejection $A_\epsilon^{\text{rej}}(f, \alpha)$ for a given $\epsilon > 0$ and $\alpha \in [0, 1]$, we need to generate two adversarial examples $\mathbf{x}' \in \mathcal{N}(\mathbf{x}, \alpha\epsilon)$ and $\mathbf{x}'' \in \mathcal{N}(\mathbf{x}, \epsilon)$ for each clean input $(\mathbf{x}, y)$. We call the attack for generating $\mathbf{x}' \in \mathcal{N}(\mathbf{x}, \alpha\epsilon)$ the inner attack, and the attack for generating $\mathbf{x}'' \in \mathcal{N}(\mathbf{x}, \epsilon)$ the outer attack.

For both the inner attack and outer attack, we use an ensemble of attacks and report the worst-case robustness. For the inner attack, if any of the attacks in the inner-attack ensemble finds an adversarial example $\mathbf{x}' \in \mathcal{N}(\mathbf{x}, \alpha\epsilon)$ that is rejected by the model, then we consider the inner attack to be successful on the clean input

$(\mathbf{x}, y)$. For the outer attack, if any of the attacks in the outer-attack ensemble finds an adversarial example $\mathbf{x}'' \in \mathcal{N}(\mathbf{x}, \epsilon)$ that is accepted and misclassified by the model, then we consider the outer attack to be successful on the clean input $(\mathbf{x}, y)$.

In this section, we design adaptive attacks to generate $\mathbf{x}'$ and $\mathbf{x}''$ for the different methods using the same underlying principles to ensure fair comparison. The following ideas are applied to design the attack loss functions for all the methods. Whenever the attack objective is to maximize a probability (or a probability-like) term $p \in [0, 1]$, we use the loss function $\log(p) \in (-\infty, 0]$. Similarly, the loss function $-\log(p) \in [0, \infty)$ is used to minimize a probability (or a probability-like) term p.

We first introduce the attack objectives for all the methods below, and then discuss how we solve the attack objectives.

A.3.1 Attack Objectives

In this section, we describe the attack objectives used to generate the adversarial examples $\mathbf{x}' \in \mathcal{N}(\mathbf{x}, \alpha\epsilon)$ (i.e., the inner attack) and the adversarial examples $\mathbf{x}'' \in \mathcal{N}(\mathbf{x}, \epsilon)$ (i.e., the outer attack) from a clean input $(\mathbf{x}, y)$ for the different methods compared. The goal of the adversary to generate $\mathbf{x}'$ is to make the defense method reject $\mathbf{x}'$. The goal of the adversary to generate $\mathbf{x}''$ is to make the defense method accept and incorrectly classify $\mathbf{x}''$ into a class other than y. We next discuss the inner and outer attack objectives for the different methods considered.

Confidence-based Rejection. The methods AT+CR, TRADES+CR and CCAT use the classifier's confidence (i.e., maximum softmax probability) as the score for rejection. Suppose the softmax output of the classifier is

$$\mathbf{h}(\mathbf{x}; \theta) = [h_1(\mathbf{x}; \theta), \cdots, h_k(\mathbf{x}; \theta)],$$

then the score for acceptance is the confidence given by $h_{\max}(\mathbf{x}; \theta) = \max_j h_j(\mathbf{x}; \theta)$. We use the log-sum-exp approximation of the max function to define a smooth inner attack objective that minimizes the confidence score $h_{\max}(\mathbf{x}; \theta)$. We use the

fact that

$$\frac{1}{\tau}\log\Big(\sum_{i=1}^{k} e^{\tau s_i}\Big) \;\approx\; \max_{i\in[k]} s_i \tag{A.12}$$

where the approximation becomes better for larger values of the temperature constant $\tau > 0$. We would like to approximate the exact inner attack objective given by

$$\mathbf{x}' \;=\; \arg\max_{\mathbf{z}\in\mathcal{N}(\mathbf{x},\alpha\epsilon)} -\log h_{\max}(\mathbf{z};\theta). \tag{A.13}$$

This attack aims to find an adversarial input $\mathbf{x}'$ that minimizes the confidence, thus causing the input to be rejected by methods using confidence-based rejection. Let $\widetilde{\mathbf{h}}(\mathbf{x};\theta) = [\widetilde{h}_1(\mathbf{x};\theta), \cdots, \widetilde{h}_k(\mathbf{x};\theta)]$ denote the logits corresponding to the classifier prediction, with $\widetilde{h}_{\max}(\mathbf{x};\theta)$ being the maximum logit. The attack objective (A.13) can be approximated as

$$\begin{aligned}
\mathbf{x}' \;&=\; \arg\max_{\mathbf{z}\in\mathcal{N}(\mathbf{x},\alpha\epsilon)} -\log h_{\max}(\mathbf{z};\theta) \\
&=\; \arg\max_{\mathbf{z}\in\mathcal{N}(\mathbf{x},\alpha\epsilon)} -\widetilde{h}_{\max}(\mathbf{z};\theta) \;+\; \log\Big(\sum_{i=1}^{k} e^{\widetilde{h}_i(\mathbf{z};\theta)}\Big) \\
&\approx\; \arg\max_{\mathbf{z}\in\mathcal{N}(\mathbf{x},\alpha\epsilon)} -\frac{1}{\tau}\log\Big(\sum_{i=1}^{k} e^{\tau\widetilde{h}_i(\mathbf{z};\theta)}\Big) \;+\; \log\Big(\sum_{i=1}^{k} e^{\widetilde{h}_i(\mathbf{z};\theta)}\Big).
\end{aligned} \tag{A.14}$$

We name this attack with objective (A.14) Low Confidence Inner Attack (**LCIA**). In our experiments, we set $\tau = 100$.

For the outer attack, we use the multi-targeted PGD approach. Specifically, for each target label $j \neq y$, we generate an adversarial example $\mathbf{x}''_j \in \mathcal{N}(\mathbf{x}, \epsilon)$ via the following objective:

$$\mathbf{x}''_j \;=\; \arg\max_{\mathbf{z}\in\mathcal{N}(\mathbf{x},\epsilon)} \log h_j(\mathbf{z};\theta). \tag{A.15}$$

Then we select the strongest adversarial example $\mathbf{x}''$ via:

$$\mathbf{x}'' = \mathbf{x}''_{j^\star} \quad \text{s.t.} \quad j^\star = \underset{j \in [k] \setminus \{y\}}{\arg\max} \log h_j(\mathbf{x}''_j; \theta). \tag{A.16}$$

By solving this objective, the adversary attempts to find an adversarial example that is misclassified with high confidence. The goal of the adversary is to make the selective classifier accept and incorrectly classify the adversarial input. We name this attack with objective (A.15) High Confidence Misclassification Outer Attack (**HCMOA**). Note that this HCMOA attack is stronger than the PGD attack with backtracking proposed in Stutz et al. (2020) for evaluating the robustness of CCAT.

RCD. The RCD method (Sheikholeslami et al., 2021) trains a $(k+1)$-way classifier such that class $k+1$ is treated as the rejection class. Suppose the softmax output of the classifier is $\mathbf{h}(\mathbf{x}; \theta) = [h_1(\mathbf{x}; \theta), \cdots, h_{k+1}(\mathbf{x}; \theta)]$. For the inner attack, we generate the adversarial example $\mathbf{x}' \in \mathcal{N}(\mathbf{x}, \alpha\epsilon)$ using the following objective:

$$\mathbf{x}' = \underset{\mathbf{z} \in \mathcal{N}(\mathbf{x}, \alpha\epsilon)}{\arg\max} \log h_{k+1}(\mathbf{z}; \theta). \tag{A.17}$$

The goal of the adversary is to make the method reject the adversarial input by pushing the probability of class $k+1$ close to 1. We name this attack with objective (A.17) RCD Inner Attack (**RCDIA**).

For the outer attack, we use the multi-targeted PGD approach. Specifically, for each target label $j \notin \{y, k+1\}$, we generate the adversarial example $\mathbf{x}''_j \in \mathcal{N}(\mathbf{x}, \epsilon)$ via the following objective:

$$\mathbf{x}''_j = \underset{\mathbf{z} \in \mathcal{N}(\mathbf{x}, \epsilon)}{\arg\max} \log h_j(\mathbf{z}; \theta). \tag{A.18}$$

Then we select the strongest adversarial example $\mathbf{x}''$ via:

$$\mathbf{x}'' = \mathbf{x}''_{j^\star} \quad \text{s.t.} \quad j^\star = \underset{j \notin \{y, k+1\}}{\arg\max} \log h_j(\mathbf{x}''_j; \theta). \tag{A.19}$$

Here, the goal of the adversary is to make the selective classifier accept and incorrectly classify the adversarial input, and this objective achieves this by increasing the probability of a class that is different from both the true class y and the rejection class $k+1$ to be close to 1. We name this attack with objective (A.18) RCD Outer Attack (**RCDOA**).

ATRR. The ATRR method (Pang et al., 2022) uses a rectified confidence score for rejection. Suppose the softmax output of the classifier is

$$\mathbf{h}(\mathbf{x}\,;\theta) = [h_1(\mathbf{x}\,;\theta), \cdots, h_k(\mathbf{x}\,;\theta)]$$

and the auxiliary function is $A(\mathbf{x};\phi) \in [0,1]$. The rectified confidence is defined by Pang et al. (2022) as the product of the auxiliary function and the classifier's confidence, i.e., $A(\mathbf{x};\phi)\, h_{\max}(\mathbf{x}\,;\theta)$. For the inner attack, we generate the adversarial example $\mathbf{x}' \in \mathcal{N}(\mathbf{x}, \alpha\epsilon)$ using the following objective:

$$\begin{aligned}
\mathbf{x}' \;&=\; \underset{\mathbf{z}\in\mathcal{N}(\mathbf{x},\alpha\epsilon)}{\arg\max} - \log A(\mathbf{z};\phi) \;-\; \log h_{\max}(\mathbf{z}\,;\theta) \\
&\approx\; \underset{\mathbf{z}\in\mathcal{N}(\mathbf{x},\alpha\epsilon)}{\arg\max} - \log A(\mathbf{z};\phi) \;+\; \Big[\log\Big(\sum_{i=1}^{k} e^{\tilde{h}_i(\mathbf{z}\,;\theta)}\Big) - \frac{1}{\tau}\log\Big(\sum_{i=1}^{k} e^{\tau\tilde{h}_i(\mathbf{z}\,;\theta)}\Big)\Big].
\end{aligned} \tag{A.20}$$

Here, we use the log-sum-exp approximation of the max function $h_{\max}(\mathbf{z}\,;\theta) = \max_j h_j(\mathbf{z}\,;\theta)$ and set $\tau = 100$. The goal is to minimize the rectified confidence by either minimizing the auxiliary function (first term) or the classifier's confidence by pushing its predictions to be close to uniform (second term). By minimizing the rectified confidence score, the adversary attempts to make the ATRR method reject the perturbed input $\mathbf{x}'$. We name this attack with objective (A.20) ATRR Inner Attack (**ATRRIA**).

For the outer attack, we use the multi-targeted PGD approach. Specifically, for each target label $j \neq y$, we generate the adversarial example $\mathbf{x}_j'' \in \mathcal{N}(\mathbf{x}, \epsilon)$ via the

following objective:

$$\mathbf{x}_j'' \;=\; \underset{\mathbf{z}\in\mathcal{N}(\mathbf{x},\epsilon)}{\arg\max}\, \log\Big[A(\mathbf{z};\phi)\, h_j(\mathbf{z}\,;\theta)\Big]. \tag{A.21}$$

Then we select the strongest adversarial example $\mathbf{x}''$ via:

$$\mathbf{x}'' \;=\; \mathbf{x}''_{j^\star} \quad \text{s.t.} \quad j^\star \;=\; \underset{j\in[k]\setminus\{y\}}{\arg\max}\, \log\Big[A(\mathbf{x}_j'';\phi)\, h_j(\mathbf{x}_j''\,;\theta)\Big]. \tag{A.22}$$

The goal of the adversary is to make the selective classifier accept and incorrectly classify the adversarial input. The objective achieves this by pushing the rectified confidence as well as the predicted probability of a class different from y close to 1. This ensures that adversarial input is accepted as well as incorrectly classified. We name this attack with objective (A.21) ATRR Outer Attack (**ATRROA**).

CPR (proposed defense). The goal of the inner attack for CPR is to find $\mathbf{x}' \in \mathcal{N}(\mathbf{x}, \alpha\epsilon)$ such that the base model has different predictions on $\mathbf{x}'$ and $T(\mathbf{x}')$, thus ensuring rejection. We consider three adaptive inner attacks that can achieve this goal. The first one is the Low-Confidence Inner Attack (**LCIA**) introduced in Section 3.5.2. This attack aims to find $\mathbf{x}' \in \mathcal{N}(\mathbf{x}, \alpha\epsilon)$ where the base model has low confidence. Recall that the mapping $T(\mathbf{x}')$ attempts to minimize the base model's probability on the predicted class $\widehat{y}(\mathbf{x}')$. So, if the base model has low confidence on $\mathbf{x}'$, then it will very likely have even lower probability for $\widehat{y}(\mathbf{x}')$ on $T(\mathbf{x}')$, and thus have different predictions on $T(\mathbf{x}')$ and $\mathbf{x}'$.

The second inner attack is a variant of LCIA, named Consistent-Low-Confidence Inner Attack (**CLCIA**), which attempts to find an adversarial example $\mathbf{x}'$ by minimizing the confidence of the base model on both $\mathbf{x}'$ and $T(\mathbf{x}')$. The CLCIA attack

has the following objective:

$$\begin{aligned}
\mathbf{x}' \;=\; & \underset{\mathbf{z}\in\mathcal{N}(\mathbf{x},\alpha\epsilon)}{\arg\max}\Big[-\log h_{\max}(\mathbf{z};\theta) \;-\; \log h_{\max}(T(\mathbf{z});\theta)\Big] \\
\approx\; & \underset{\mathbf{z}\in\mathcal{N}(\mathbf{x},\alpha\epsilon)}{\arg\max}\Big[-\frac{1}{\tau}\log\Big(\sum_{i=1}^{k} e^{\tau\widetilde{h}_i(\mathbf{z};\theta)}\Big) \;+\; \log\Big(\sum_{i=1}^{k} e^{\widetilde{h}_i(\mathbf{z};\theta)}\Big) \\
& \qquad -\frac{1}{\tau}\log\Big(\sum_{i=1}^{k} e^{\tau\widetilde{h}_i(T(\mathbf{z});\theta)}\Big) \;+\; \log\Big(\sum_{i=1}^{k} e^{\widetilde{h}_i(T(\mathbf{z});\theta)}\Big)\Big]. \qquad \text{(A.23)}
\end{aligned}$$

We have denoted the logits of the base classifier by

$$\widetilde{\mathbf{h}}(\mathbf{x};\theta) = [\widetilde{h}_1(\mathbf{x};\theta),\cdots,\widetilde{h}_k(\mathbf{x};\theta)],$$

and we apply the log-sum-exp approximation to the max function (as before) with $\tau = 100$. We use the backward pass differentiable approximation (BPDA) method (Athalye et al., 2018) to solve this objective since $T(\mathbf{x})$ does not have a closed-form expression and is not differentiable.

The third inner attack is a multi-targeted attack, also based on BPDA, which considers all possible target classes and attempts to find an $\mathbf{x}'$ such that the base model has high probability for the target class at $\mathbf{x}'$ and a low probability for the target class at $T(\mathbf{x}')$ (thereby encouraging rejection). The attack objective is: for each target class $j = 1,\ldots,k$,

$$\mathbf{x}'_j \;=\; \underset{\mathbf{z}\in\mathcal{N}(\mathbf{x},\alpha\epsilon)}{\arg\max}\Big[\log h_j(\mathbf{z};\theta) \;-\; \log h_j(T(\mathbf{z});\theta)\Big]. \qquad \text{(A.24)}$$

Then we select the strongest adversarial example $\mathbf{x}'$ via:

$$\mathbf{x}' \;=\; \mathbf{x}'_{j^\star} \quad \text{s.t.} \quad j^\star \;=\; \underset{j\in[k]}{\arg\max}\Big[\log h_j(\mathbf{x}'_j;\theta) \;-\; \log h_j(T(\mathbf{x}'_j);\theta)\Big]. \qquad \text{(A.25)}$$

We name this third inner attack Prediction-Disagreement Inner Attack (**PDIA**).

Given a clean input $(\mathbf{x}, y)$, the goal of the outer attack is to find $\mathbf{x}'' \in \mathcal{N}(\mathbf{x},\epsilon)$ such that the base model has consistent wrong predictions on both $\mathbf{x}''$ and $T(\mathbf{x}'')$.

This ensures that $\mathbf{x}''$ is accepted and mis-classified. We consider two adaptive outer attacks that can achieve this goal. As discussed in Section 3.5.2, the first outer attack is a multi-targeted attack based on BPDA with the following objective: for each target class $j \in [k] \setminus \{y\}$,

$$\mathbf{x}''_j \;=\; \underset{\mathbf{z}\in\mathcal{N}(\mathbf{x},\epsilon)}{\arg\max} \Big[\log h_j(\mathbf{z}\,;\theta) \;+\; \log h_j(T(\mathbf{z})\,;\theta) \Big]. \tag{A.26}$$

Then we select the strongest adversarial example $\mathbf{x}''$ via:

$$\mathbf{x}'' \;=\; \mathbf{x}''_{j^\star} \quad \text{s.t.} \quad j^\star \;=\; \underset{j\in[k]\setminus\{y\}}{\arg\max} \Big[\log h_j(\mathbf{x}''_j\,;\theta) \;+\; \log h_j(T(\mathbf{x}''_j)\,;\theta) \Big]. \tag{A.27}$$

We name this outer attack Consistent High Confidence Misclassification Outer Attack (**CHCMOA**).

The second outer attack is the High Confidence Misclassification Outer Attack (**HCMOA**) that was discussed earlier for the methods based on confidence-based rejection. The attack objective is given in Eqns. (A.15) and (A.16). The intuition for this attack is that if the base model has a high-confidence incorrect prediction on $\mathbf{x}''$, then it becomes hard for T to change the incorrect prediction.

A.3.2 Solving the Attack Objectives

We use the PGD with momentum to solve the attack objectives, and use the PGD attack with multiple restarts for evaluating the robustness. Following Stutz et al. (2020), we initialize the perturbation δ uniformly over directions and norm as follows:

$$\delta \;=\; u\,\epsilon\,\frac{\delta'}{\|\delta'\|_\infty},\; \delta' \sim \mathcal{N}(\mathbf{0},\mathbf{I}),\; u \sim U(0,1) \tag{A.28}$$

where δ' is sampled from the standard Gaussian and $u \in [0,1]$ is sampled from the uniform distribution. We also include zero initialization, i.e., $\delta = \mathbf{0}$ as a candidate. We allocate one restart for zero initialization, and multiple restarts for the random

initializations. We finally select the perturbation corresponding to the best objective value obtained throughout the optimization.

When solving the inner attack objectives, we use a momentum factor of 0.9, 200 iterations, and 5 random restarts. The base learning rate (i.e., the attack step size) is varied over the set $\{0.1, 0.01, 0.005\}$ for experiments on MNIST, and over the set $\{0.01, 0.005, 0.001\}$ for experiments on SVHN and CIFAR-10. We report the worst-case results: for each clean input $\mathbf{x}$, if the PGD method with a particular base learning rate can find an $\mathbf{x}'$ that is rejected, then we will use this $\mathbf{x}'$ as the generated adversarial example and consider the inner attack to be successful.

When solving the outer attack objectives, we use a momentum factor of 0.9, 200 iterations, and 5 random restarts. The base learning rate (i.e., the attack step size) is varied over the set $\{0.1, 0.01, 0.005\}$ for experiments on MNIST, and over the set $\{0.01, 0.005, 0.001\}$ for experiments on SVHN and CIFAR-10. We report the worst-case results: for each clean input $\mathbf{x}$, if the PGD method with a particular base learning rate can find an $\mathbf{x}''$ that is accepted and misclassified, then we will use this $\mathbf{x}''$ as the generated adversarial example and consider the outer attack to be successful.

BPDA approximation. Three of the adaptive attacks for CPR (namely CLCIA, PDIA, and CHCMOA) depend on $T(\mathbf{x})$, which does not have a closed-form expression and is not differentiable. Therefore, it is not possible to calculate the exact gradient for these attack objectives. We will use the BPDA approach (Athalye et al., 2018) to address this challenge, specifically using the straight-through estimator for the gradient.

Recall that $T(\mathbf{x}) \approx \arg\max_{\widetilde{\mathbf{x}} \in \mathcal{N}(\mathbf{x}, \widetilde{\epsilon})} \ell_{CE}(\widetilde{\mathbf{x}}, \widehat{y}(\mathbf{x}))$, and it is computed using Algorithm 3. For any of these attack objectives, let $\ell(T(\mathbf{x}))$ denote the terms dependent on $T(\cdot)$. For instance, $\ell(T(\mathbf{x})) = \log h_j(T(\mathbf{x}); \theta)$ for the CHCMOA attack. Using the chain rule, we can express the gradient of ℓ as follows:

$$\nabla_{\mathbf{x}} \ell(T(\mathbf{x})) \;=\; \mathbf{J}_T(\mathbf{x})^t \, \nabla_{\mathbf{u}} \ell(\mathbf{u})\big|_{\mathbf{u}=T(\mathbf{x})} \tag{A.29}$$

where $\mathbf{J}_T(\mathbf{x})^t$ is the transpose of the $d \times d$ Jacobian of $T(\mathbf{x})$. For a small $\widetilde{\epsilon}$, we make the approximation that $T(\mathbf{x}) \approx \mathbf{x}$ during the backward pass, which in turn makes $\mathbf{J}_T(\mathbf{x})$ approximately equal to the identity matrix. This gives the following gradient estimate of the BPDA method, which we apply to solve the attack objectives of CLCIA, PDIA, and CHCMOA:

$$\nabla_{\mathbf{x}} \ell(T(\mathbf{x})) \;=\; \nabla_{\mathbf{u}} \ell(\mathbf{u})\big|_{\mathbf{u}=T(\mathbf{x})} \tag{A.30}$$

During the forward pass, we perform an exact computation of $T(x)$ (using Algorithm 3), but during the backward pass, we approximate the gradient of the attack objectives using Eqn.(A.30). We note that these adaptive attacks are more expensive to run because they require the computation of $T(\mathbf{x})$ during each PGD step.

A.3.3 Attack Ensemble

As discussed earlier, for each defense method, we consider an ensemble of inner and outer attacks and report the worst-case robustness with rejection under these attacks. We list the specific attacks in the ensemble for each defense method below:

Confidence-based Rejection. The inner-attack ensemble only the includes Low Confidence Inner Attack (LCIA). The outer-attack ensemble includes the AutoAttack (Croce and Hein, 2020) and High Confidence Misclassification Outer Attack (HCMOA).

RCD. The inner-attack ensemble only includes RCDIA. The outer-attack ensemble includes AutoAttack and RCDOA.

ATRR. The inner-attack ensemble only includes ATRRIA. The outer-attack ensemble includes AutoAttack and ATRROA.

CPR (proposed). The inner-attack ensemble includes LCIA, CLCIA, and PDIA. The outer-attack ensemble includes AutoAttack, HCMOA and CHCMOA. By in-

cluding a number of strong attacks in the ensemble, we have attempted to perform a thorough evaluation of CPR.

A.3.4 Evaluating Adaptive Attacks for CPR

In Section 3.6.2, we perform an ablation study to compare the strength of the different adaptive inner and outer attacks for CPR. These results are in Table 3.2, and here we discuss the choice of metrics for this evaluation. The outer attack only affects $A_\epsilon^{\text{rej}}(f, 0)$, while the inner attack affects $A_\epsilon^{\text{rej}}(f, \alpha)$ for $\alpha > 0$. Therefore, for the outer attacks we only need to compare $A_\epsilon^{\text{rej}}(f, 0)$. For the inner attacks we compare $A_\epsilon^{\text{rej}}(f, 1)$, while fixing the outer attack to be the strongest ensemble outer attack. This corresponds to the right end of the robustness curve, and gives a clear idea of the strength of the inner attack.

B APPENDIX FOR CHAPTER 4

B.1 Proofs

B.1.1 Additional definitions

Let P, Q be two distributions, a coupling $M = (Z, Z')$ is a joint distribution, where, if we marginalize M to the first component, Z, it is identically distributed as P, and if we marginalize M to the second component, Z', it is identically distributed as Q. Let $\prod(P, Q)$ be the set of all couplings of P and Q, and let $c(\cdot, \cdot)$ be a "cost" function that maps (z, z') to a real value. Wasserstein distance between P and Q w.r.t. c is defined as

$$W_c(P, Q) = \inf_{M \in \prod(P,Q)} \left\{ \mathop{\mathbb{E}}_{(z,z') \sim M} [c(z, z')] \right\}.$$

Intuitively, this is to find the "best transportation plan" (a coupling M) to minimize the expected transportation cost (transporting z to z' where the cost is $c(z, z')$).

B.1.2 Integrated Gradients for an Intermediate Layer

In this section we show how to compute Integrated Gradients for an intermediate layer of a neural network. Let $h : \mathbb{R}^d \mapsto \mathbb{R}^k$ be a function that computes a hidden layer of a neural network, where we map a d-dimensional input vector to a k-dimensional output vector. Given two points $\mathbf{x}$ and $\mathbf{x}'$ for computing attribution, again we consider a parameterization (which is a mapping $r : \mathbb{R} \mapsto \mathbb{R}^d$) such that $r(0) = \mathbf{x}$, and $r(1) = \mathbf{x}'$.

The key insight is to leverage the fact that Integrated Gradients is a *curve integration*. Therefore, given some hidden layer, one can then naturally view the previous layers as inducing a *curve* $h \circ r$ which moves from $h(\mathbf{x})$ to $h(\mathbf{x}')$, as we move from $\mathbf{x}$ to $\mathbf{x}'$ along curve r. Viewed this way, we can thus naturally compute IG for $\mathbf{h}$ in a way that leverages all layers of the network. Specifically, consider another curve $\gamma(t) : \mathbb{R} \mapsto \mathbb{R}^k$, defined as $\gamma(t) = h(r(t))$, to compute a curve integral. By definition

we have $f(\mathbf{x}) = g(h(\mathbf{x}))$ and

$$\begin{aligned} f(\mathbf{x}') - f(\mathbf{x}) &= g(h(\mathbf{x}')) - g(h(\mathbf{x})) \\ &= g(\gamma(1)) - g(\gamma(0)) \\ &= \int_0^1 \sum_{i=1}^{k} \frac{\partial f(\gamma(t))}{\partial h_i} \gamma_i'(t) dt \\ &= \sum_{i=1}^{k} \int_0^1 \frac{\partial f(\gamma(t))}{\partial h_i} \gamma_i'(t) dt \end{aligned}$$

Therefore we can define the attribution of h_i naturally as

$$IG_{h_i}^f(\mathbf{x}, \mathbf{x}') = \int_0^1 \frac{\partial f(\gamma(t))}{\partial h_i} \gamma_i'(t) dt$$

Let's unpack this a little more:

$$\begin{aligned} \int_0^1 \frac{\partial f(\gamma(t))}{\partial h_i} \gamma_i'(t) dt &= \int_0^1 \frac{\partial f(h(r(t)))}{\partial h_i} \sum_{j=1}^{d} \frac{\partial h_i(r(t))}{\partial \mathbf{x}_j} r_j'(t) dt \\ &= \int_0^1 \frac{\partial f(h(r(t)))}{\partial h_i} \sum_{j=1}^{d} \frac{\partial h_i(r(t))}{\partial \mathbf{x}_j} r_j'(t) dt \\ &= \sum_{j=1}^{d} \left\{ \int_0^1 \frac{\partial f(h(r(t)))}{\partial h_i} \frac{\partial h_i(r(t))}{\partial \mathbf{x}_j} r_j'(t) dt \right\} \end{aligned}$$

This thus gives the lemma:

Lemma B.1. *Under curve* $r : \mathbb{R} \mapsto \mathbb{R}^d$ *where* $r(0) = \mathbf{x}$ *and* $r(1) = \mathbf{x}'$*, the attribution for* h_i *for a differentiable function* f *is*

$$IG_{h_i}^f(\mathbf{x}, \mathbf{x}', r) = \sum_{j=1}^{d} \left\{ \int_0^1 \frac{\partial f(h(r(t)))}{\partial h_i} \frac{\partial h_i(r(t))}{\partial \mathbf{x}_j} r_j'(t) dt \right\} \tag{B.1}$$

Note that (6) nicely recovers attributions for input layer, in which case h is the

identity function.

Summation approximation. Similarly, we can approximate the above Riemann integral using a summation. Suppose we slice $[0, 1]$ into m equal segments, then (4.2) can be approximated as:

$$\mathrm{IG}^{f}_{h_i}(\mathbf{x}, \mathbf{x}') = \frac{1}{m} \sum_{j=1}^{d} \left\{ \sum_{k=0}^{m-1} \frac{\partial f(h(r(k/m)))}{\partial h_i} \frac{\partial h_i(r(k/m))}{\partial \mathbf{x}_j} r'_j(k/m) \right\} \tag{B.2}$$

B.1.3 Proof of Proposition 1

If we put $\lambda = 1$ and let $s(\cdot)$ be the sum function (sum all components of a vector), then for any curve r and any intermediate layer $\mathbf{h}$, (4.4) becomes:

$$\begin{aligned}
\rho(\mathbf{x}, y; \theta) &= \ell(\mathbf{x}, y; \theta) + \max_{\mathbf{x}' \in N(\mathbf{x}, \varepsilon)} \{\mathrm{sum}(\mathrm{IG}^{\ell_y}(\mathbf{x}, \mathbf{x}'; r))\} \\
&= \ell(\mathbf{x}, y; \theta) + \max_{\mathbf{x}' \in N(\mathbf{x}, \varepsilon)} \{\ell(\mathbf{x}', y; \theta) - \ell(\mathbf{x}, y; \theta)\} \\
&= \max_{\mathbf{x}' \in N(\mathbf{x}, \varepsilon)} \ell(\mathbf{x}', y; \theta)
\end{aligned}$$

where the second equality is due to the Axiom of Completeness of IG.

B.1.4 Proof of Proposition 2

Input gradient regularization is an old idea proposed by Drucker and LeCun (1992), and is recently used by Ross and Doshi-Velez (2018) in adversarial training setting. Basically, for $q \geqslant 1$, they propose $\rho(\mathbf{x}, y; \theta) = \ell(\mathbf{x}, y; \theta) + \lambda \|\nabla_x \ell(\mathbf{x}, y; \theta)\|_q^q$, where they want small gradient at $\mathbf{x}$. To recover this objective from robust attribution regularization, let us pick $s(\cdot)$ as the $\|\cdot\|_1^q$ function (1-norm to the q-th power), and consider the simplest curve $r(t) = \mathbf{x} + t(\mathbf{x}' - \mathbf{x})$. With the naïve summation approximation of the integral $\mathrm{IG}_i^{\ell_y}$ we have $\mathrm{IG}_i^{\ell_y}(\mathbf{x}, \mathbf{x}'; r) \approx \frac{(\mathbf{x}'_i - \mathbf{x}_i)}{m} \sum_{k=1}^{m} \frac{\partial \ell(\mathbf{x} + \frac{k-1}{m}(\mathbf{x}' - \mathbf{x}), y; \theta)}{\partial \mathbf{x}_i}$, where larger m is, more accurate we approximate the integral. Now, if we put $m = 1$, which is the coarsest approximation, this becomes $(\mathbf{x}'_i - \mathbf{x}_i) \frac{\partial \ell(\mathbf{x}, y; \theta)}{\partial \mathbf{x}_i}$, and we

have $\mathrm{IG}^{\ell_y}(\mathbf{x}, \mathbf{x}'; \theta) = (\mathbf{x}' - \mathbf{x}) \odot \nabla_{\mathbf{x}} \ell(\mathbf{x}, y; \theta)$. Therefore (4.4) becomes:

$$\begin{aligned}\rho(\mathbf{x}, y; \theta) =& \ell(\mathbf{x}, y; \theta) + \lambda \max_{\mathbf{x}' \in \mathsf{N}(\mathbf{x}, \varepsilon)} \{\| \mathrm{IG}^{\ell_y}(\mathbf{x}, \mathbf{x}'; \theta)\|_1^q\} \\ \approx& \ell(\mathbf{x}, y; \theta) + \lambda \max_{\mathbf{x}' \in \mathsf{N}(\mathbf{x}, \varepsilon)} \{\|(\mathbf{x}' - \mathbf{x}) \odot \nabla_{\mathbf{x}} \ell(\mathbf{x}, y; \theta)\|_1^q\}\end{aligned}$$

Put the neighborhood as $\|\mathbf{x}' - \mathbf{x}\|_p \leqslant \varepsilon$ where $p \in [1, \infty]$ and $\frac{1}{p} + \frac{1}{q} = 1$. By Hölder's inequality, $\|(\mathbf{x}' - \mathbf{x}) \odot \nabla_{\mathbf{x}} \ell(\mathbf{x}, y; \theta)\|_1^q \leqslant \|\mathbf{x}' - \mathbf{x}\|_p^q \|\nabla \ell(\mathbf{x}, y; \theta)\|_q^q \leqslant \varepsilon^q \|\nabla \ell(\mathbf{x}, y; \theta)\|_q^q$ which means that $\max_{\|\mathbf{x}' - \mathbf{x}\|_p \leqslant \varepsilon} \{\|(\mathbf{x}' - \mathbf{x}) \odot \nabla_{\mathbf{x}} \ell(\mathbf{x}, y; \theta)\|_1^q\} = \varepsilon^q \|\nabla \ell(\mathbf{x}, y; \theta)\|_q^q$. Thus by putting $\lambda = \lambda'/\varepsilon^q$, we recover gradient regularization with regularization parameter λ'.

B.1.5 Proof of Proposition 3

Let us put $s(\cdot) = \|\cdot\|_1$, and $\mathbf{h} = \ell_y$ (the output layer of loss function!), then we have

$$\begin{aligned}\rho(\mathbf{x}, y; \theta) =& \ell_y(\mathbf{x}) + \max_{\mathbf{x}' \in \mathsf{N}(\mathbf{x}, \varepsilon)} \{\| \mathrm{IG}_{\ell_y}^{\ell_y}(\mathbf{x}, \mathbf{x}'; r)\|_1\} \\ =& \ell_y(\mathbf{x}) + \max_{\mathbf{x}' \in \mathsf{N}(\mathbf{x}, \varepsilon)} \{|\ell_y(\mathbf{x}') - \ell_y(\mathbf{x})|\}\end{aligned}$$

where the second equality is because $\mathrm{IG}_{\ell_y}^{\ell_y}(\mathbf{x}, \mathbf{x}'; r) = \ell_y(\mathbf{x}') - \ell_y(\mathbf{x})$.

B.1.6 Proof of Proposition 4

Specifically, again, let $s(\cdot)$ be the summation function and $\lambda = 1$, then we have $\mathbb{E}_{Z,Z'}[d_{\mathrm{IG}}(Z, Z')] = \mathbb{E}_{Z,Z'}[\mathrm{sum}(\mathrm{IG}_{\mathbf{h}}^{\ell}(Z, Z'))] = \mathbb{E}_{Z,Z'}[\ell(Z'; \theta) - \ell(Z; \theta)]$. Because P

and Z are identically distributed, thus the objective reduces to

$$\begin{aligned}
&\sup_{Q;M\in\prod(P,Q)} \Big\{ \mathop{\mathbb{E}}_{Z,Z'}[\ell(Z;\theta)+\ell(Z';\theta)-\ell(Z;\theta)] \\
&\qquad\qquad \text{s.t. } \mathop{\mathbb{E}}_{Z,Z'}[c(Z,Z')] \leqslant \rho \Big\} \\
&= \sup_{Q;M\in\prod(P,Q)} \left\{ \mathop{\mathbb{E}}_{Z'}[\ell(Z';\theta)] \text{ s.t. } \mathop{\mathbb{E}}_{Z,Z'}[c(Z,Z')] \leqslant \rho \right\} \\
&= \sup_{Q:W_c(P,Q)\leqslant\rho} \left\{ \mathop{\mathbb{E}}_{Q}[\ell(Q;\theta)] \right\},
\end{aligned}$$

which is exactly Wasserstein prediction robustness objective.

B.1.7 Proof of Theorem 4.2

The proof largely follows that for Theorem 5 in Sinha et al. (2018), and we provide it here for completeness. Since we have a joint supremum over Q and $M \in \prod(P,Q)$ we have that

$$\begin{aligned}
\sup_{Q;M\in\prod(P,Q)} \left\{ \mathop{\mathbb{E}}_{M=(Z,Z')} \left[d_{IG}^{\gamma}(Z,Z') \right] \right\} &= \sup_{Q;M\in\prod(P,Q)} \int [d_{IG}(z,z') - \gamma c(z,z')] dM(z,z') \\
&\leqslant \int \sup_{z'} \{ d_{IG}(z,z') - \gamma c(z,z') \} dP(z) \\
&= \mathop{\mathbb{E}}_{z\sim P} \left[\sup_{z'} \{ d_{IG}^{\gamma}(z,z') \} \right].
\end{aligned}$$

We would like to show equality in the above.

Let $\mathcal{Q}$ denote the space of regular conditional probabilities from Z to Z'. Then

$$\begin{aligned}
&\sup_{Q;M\in\prod(P,Q)} \int [d_{IG}(z,z') - \gamma c(z,z')] dM(z,z') \\
&\geqslant \sup_{Q\in\mathcal{Q}} \int [d_{IG}(z,z') - \gamma c(z,z')] dQ(z'|z) dP(z).
\end{aligned}$$

Let $\mathcal{Z}'$ denote all measurable mappings $z \to z'(z)$ from Z to Z'. Using the measura-

bility result in Theorem 14.60 in Rockafellar and Wets (2009), we have

$$\sup_{z' \in \mathcal{Z}'} \int [d_{IG}(z, z'(z)) - \gamma c(z, z'(z))] dP(z) = \int \sup_{z'} [d_{IG}(z, z') - \gamma c(z, z')] dP(z)$$

since $\gamma c - d_{IG}$ is a normal integrand.

Let $z'(z)$ be any measurable function that is ϵ-close to attaining the supremum above, and define the conditional distribution $Q(z'|z)$ to be supported on $z'(z)$. Then

$$\begin{aligned}
&\sup_{Q; M \in \prod(P,Q)} \int [d_{IG}(z, z') - \gamma c(z, z')] dM(z, z') \\
&\geqslant \int [d_{IG}(z, z') - \gamma c(z, z')] dQ(z'|z) dP(z) \\
&= \int [d_{IG}(z, z'(z)) - \gamma c(z, z'(z))] dP(z) \\
&\geqslant \int \sup_{z'} [d_{IG}(z, z') - \gamma c(z, z')] dP(z) - \epsilon \\
&\geqslant \sup_{Q; M \in \prod(P,Q)} \int [d_{IG}(z, z') - \gamma c(z, z')] dM(z, z') - \epsilon.
\end{aligned}$$

Since $\epsilon \geqslant 0$ is arbitrary, this completes the proof. □

B.1.8 Proof of Theorem 4.3: Connections Between the Distributional Robustness Objectives

Let θ^* denote an optimal solution of (4.5) and let θ' be any non-optimal solution. Let $\gamma(\theta^*)$ denote the corresponding γ by Lemma B.2, and $\gamma(\theta')$ denote that for θ'.

Since $\gamma(\theta')$ achieves the infimum, we have

$$\mathop{\mathbb{E}}_{z\sim P}\left[\ell(z;\theta') + \lambda \sup_{z'}\{d_{IG}(z,z') - \gamma(\theta^*)c(z,z')\}\right] \tag{B.3}$$

$$\geqslant \mathop{\mathbb{E}}_{z\sim P}\left[\ell(z;\theta') + \lambda \sup_{z'}\{d_{IG}(z,z') - \gamma(\theta')c(z,z')\}\right] \tag{B.4}$$

$$> \mathop{\mathbb{E}}_{z\sim P}\left[\ell(z;\theta^*) + \lambda \sup_{z'}\{d_{IG}(z,z') - \gamma(\theta^*)c(z,z')\}\right]. \tag{B.5}$$

So θ' is not optimal for (4.7). This then completes the proof. □

Lemma B.2. *Suppose* $c(z,z) = 0$ *and* $d_{IG}(z,z) = 0$ *for any* z*, and suppose* $\gamma c(z,z') - d_{IG}(z,z')$ *is a normal integrand. For any* $\rho > 0$*, there exists* $\gamma \geqslant 0$ *such that*

$$\sup_{Q;M\in\prod(P,Q)}\left\{\mathop{\mathbb{E}}_{(Z,Z')\sim M}[d_{IG}(Z,Z')] \text{ s.t. } \mathop{\mathbb{E}}_{(Z,Z')\sim M}[c(Z,Z')] \leqslant \rho\right\} \tag{B.6}$$

$$= \inf_{\zeta\geqslant 0}\mathop{\mathbb{E}}_{z\sim P}\left[\sup_{z'}\{d_{IG}(z,z') - \zeta c(z,z') + \zeta\rho\}\right]. \tag{B.7}$$

Furthermore, there exists $\gamma \geqslant 0$ *achieving the infimum.*

This lemma generalizes Theorem 5 in Sinha et al. (2018) to a larger, but natural, class of objectives.

Proof. For Q and $M \in \Pi(P,Q)$, let

$$\Lambda_{IG}(Q,M) := \mathop{\mathbb{E}}_{(Z,Z')\sim M}[d_{IG}(Z,Z')] \tag{B.8}$$

$$\Lambda_c(Q,M) := \mathop{\mathbb{E}}_{(Z,Z')\sim M}[c(Z,Z')] \tag{B.9}$$

First, the pair (Q,M) forms a convex set, and $\Lambda_{IG}(Q,M)$ and $\Lambda_c(Q,M)$ are linear functionals over the convex set. Set $Q = P$ and set M to the identity coupling (such that $(Z,Z') \sim M$ always has $Z = Z'$). Then $\Lambda_c(Q,M) = 0 < \rho$ and thus the Slater's condition holds. Applying standard infinite dimensional duality results (Theorem

8.6.1 in Luenberger (1997)) leads to

$$\sup_{Q;M\in\prod(P,Q);\Lambda_c(Q,M)\leqslant\rho} \Lambda_{IG}(Q,M) \tag{B.10}$$

$$= \sup_{Q;M\in\prod(P,Q)} \inf_{\zeta\geqslant 0} \{\Lambda_{IG}(Q,M) - \zeta\Lambda_c(Q,M) + \zeta\rho\} \tag{B.11}$$

$$= \inf_{\zeta\geqslant 0} \sup_{Q;M\in\prod(P,Q)} \{\Lambda_{IG}(Q,M) - \zeta\Lambda_c(Q,M) + \zeta\rho\}. \tag{B.12}$$

Furthermore, there exists $\gamma \geqslant 0$ achieving the infimum in the last line.

Now, it suffices to show that

$$\sup_{Q;M\in\prod(P,Q)} \{\Lambda_{IG}(Q,M) - \gamma\Lambda_c(Q,M) + \gamma\rho\} \tag{B.13}$$

$$= \mathop{\mathbb{E}}_{z\sim P}\left[\sup_{z'}\{d_{IG}(z,z') - \gamma c(z,z') + \gamma\rho\}\right]. \tag{B.14}$$

This is exactly what Theorem 4.2 shows.

□

B.1.9 Proof of Theorem 4.4

Let us fix any one point $\mathbf{x}$, and consider

$$g(-y_i\langle \mathbf{w}, \mathbf{x}\rangle) + \lambda \max_{\mathbf{x}'\in N(\mathbf{x},\varepsilon)} \| IG_{\mathbf{x}}^{\ell_y}(\mathbf{x},\mathbf{x}';\mathbf{w})\|_1.$$

Due to the special form of g, we know that:

$$IG_i^{\ell_y}(\mathbf{x},\mathbf{x}';\mathbf{w}) = \frac{\mathbf{w}_i(\mathbf{x}'-\mathbf{x})_i}{\langle \mathbf{w}, \mathbf{x}'-\mathbf{x}\rangle} \cdot \big(g(-y\langle \mathbf{w}, \mathbf{x}'\rangle) - g(-y\langle \mathbf{w}, \mathbf{x}\rangle)\big)$$

Let $\Delta = \mathbf{x}' - \mathbf{x}$ (which satisfies that $\|\Delta\|_\infty \leqslant \varepsilon$), therefore its absolute value (note that we are taking 1-norm):

$$\frac{\left|g(-y\langle \mathbf{w}, \mathbf{x}\rangle - y\langle \mathbf{w}, \Delta\rangle) - g(-y\langle \mathbf{w}, \mathbf{x}\rangle)\right|)}{|\langle \mathbf{w}, \Delta\rangle|} \cdot |\mathbf{w}_i \Delta_i|$$

Let $z = -y\langle \mathbf{w}, \mathbf{x}\rangle$ and $\delta = -y\langle \mathbf{w}, \Delta\rangle$, this is further simplified as $\frac{|g(z+\delta)-g(z)|}{|\delta|}|\delta_i|$. Because g is non-decreasing, so $g' \geqslant 0$, and so this is indeed $\frac{g(z+\delta)-g(z)}{\delta}$, which is the slope of the secant from $(z, g(z))$ to $(z+\delta, g(z+\delta))$. Because g is convex so the secant slopes are non-decreasing, so we can simply pick $\Delta_i = -y \operatorname{sgn}(\mathbf{w}_i)\varepsilon$, and so $\delta = \|\mathbf{w}\|_1 \varepsilon$, and so that $\| \operatorname{IG} \|_1$ becomes

$$\begin{aligned}
|g(z + \varepsilon\|\mathbf{w}\|_1) - g(z)| \cdot \frac{\sum_i |\mathbf{w}_i \Delta_i|}{|\delta|} &= |g(z + \varepsilon\|\mathbf{w}\|_1) - g(z)| \cdot \frac{\sum_i |\mathbf{w}_i|\varepsilon}{\|\mathbf{w}\|_1 \varepsilon} \\
&= |g(z + \varepsilon\|\mathbf{w}\|_1) - g(z)| \\
&= g(z + \varepsilon\|\mathbf{w}\|_1) - g(z)
\end{aligned}$$

where the last equality follows because g is nondecreasing. Therefore the objective simplifies to $\sum_{i=1}^m g(-y_i\langle \mathbf{w}, \mathbf{x}_i\rangle + \varepsilon\|\mathbf{w}\|_1)$, which is exactly Madry et al.'s objective under ℓ_∞ perturbations. □

Let us consider two examples:

Logistic Regression. Let $g(z) = \ln(1 + \exp(z))$. Then $g(-y\langle \mathbf{w}, \mathbf{x}\rangle)$ recovers the Negative Log-Likelihood loss for logistic regression. Clearly g is nondecreasing and g' is also nondecreasing. As a result, adversarial training for logistic regression is exactly "robustifying" attributions/explanations.

Softplus hinge loss. Alternatively, we can let $g(z) = \ln(1 + \exp(1 + z))$, and therefore $g(-y\langle \mathbf{w}, \mathbf{x}\rangle) = \ln(1 + \exp(1 - y\langle \mathbf{w}, \mathbf{x}\rangle))$ is the softplus version of the hinge loss function. Clearly this g also satisfy our requirements, and therefore adversarial training for softplus hinge loss function is also exactly about "robustifying" attributions/explanations.

C APPENDIX FOR CHAPTER 5

C.1 Theoretical Analysis

We consider the following data model to demonstrate how selecting informative outliers can help. In the proof below, we will use C, c to denote some absolute constants; their values can change from line to line.

1. The in-distribution P_X is a Gaussian $\mathcal{N}(\mu, \sigma^2 I)$ with mean $\mu \in \mathbb{R}^d$ and variance σ^2.

2. The test OOD distribution can be any distribution largely supported outside a ball around μ. More precisely, it can be any distribution from the family:

$$\mathcal{Q} = \left\{ Q_X : \Pr_{x \sim Q_X} [\|x - \mu\|_2 \leqslant \tau] \leqslant \epsilon_\tau \right\}$$

   where $\tau = \sigma\sqrt{d} + \sigma\gamma + \epsilon\sqrt{d}$, $\gamma \in (0, \sqrt{d})$ is a parameter indicating some margin between the in-distribution and OOD distributions, and ϵ_τ is a small number bounding the probability mass the OOD distribution can have close to the in-distribution.

3. The hypothesis class for detectors is

$$\mathcal{G} = \left\{ G_{u,r}(x) : G_{u,r}(x) = 2 \cdot \mathbb{I}[\|x - u\|_2 \leqslant r] - 1, u \in \mathbb{R}^d, r \in \mathbb{R}_+ \right\}.$$

Recall that we consider ℓ_∞ attack with adversarial budget $\epsilon > 0$. For a detector G and test OOD distribution family $\mathcal{Q}$, we are interested in the False Negative Rate $\mathrm{FNR}(G)$ and worst False Positive Rate $\sup_{Q_X \in \mathcal{Q}} \mathrm{FPR}(G; Q_X, \Omega_{\infty,\epsilon}(x))$ over $Q_X \in \mathcal{Q}$ under ℓ_∞ perturbations of magnitude ϵ. For simplicity, we denote them as $\mathrm{FNR}(G)$ and $\mathrm{FPR}(G; \mathcal{Q})$ in our proofs.

We note that the data model is set up such that there exists a robust OOD detector with good FPR and FNR (Proposition 9), while using only in-distrbution

data to learn a good robust OOD detector one needs sufficiently amount of them, i.e., $\tilde{\Theta}(d/\gamma^2)$ (Propostion 10). (Therefore, we assume $\gamma < \sqrt{d}$ to avoid the trivial case.)

Proposition 9. *The detector* $G_{u,r}(x)$ *with* $u = \mu$ *and* $r = \sigma\sqrt{d} + \sigma\gamma$ *satisfies:*

$$\mathrm{FNR}(G_{u,r}) \leqslant \exp(-c\gamma^2), \tag{C.1}$$

$$\mathrm{FPR}(G_{u,r}; \mathcal{Q}) \leqslant \epsilon_\tau, \tag{C.2}$$

for some absolute constant $c > 0$.

Proof. The first statement follows from the concentration of the norm of $x - \mu$:

$$\begin{aligned}
\mathrm{FNR}(G_{u,r}(x)) &= \mathbb{E}_{x\sim P_X}\mathbb{I}[\|x-u\|_2 > r] && \text{(C.3)}\\
&= \Pr_{x\sim\mathcal{N}(0,I)}[\|x\|_2 > \sqrt{d}+\gamma] && \text{(C.4)}\\
&\leqslant \exp(-c\gamma^2) && \text{(C.5)}
\end{aligned}$$

the second statement follows from the definition of $\mathcal{Q}$:

$$\begin{aligned}
\mathrm{FPR}(G_{u,r}; \mathcal{Q}) &= \sup_{Q_X\in\mathcal{Q}} \mathbb{E}_{x\sim Q_X} \max_{\|\delta\|_\infty\leqslant\epsilon} \mathbb{I}[\|x+\delta-u\|_2 \leqslant r] && \text{(C.6)}\\
&\leqslant \sup_{Q_X\in\mathcal{Q}} \mathbb{E}_{x\sim Q_X}\mathbb{I}[\|x-\mu\|_2 - \sqrt{d}\epsilon \leqslant \sigma\sqrt{d}+\sigma\gamma] && \text{(C.7)}\\
&= \sup_{Q_X\in\mathcal{Q}} \Pr_{x\sim Q_X}[\|x-\mu\|_2 \leqslant \tau] \leqslant \epsilon_\tau. && \text{(C.8)}
\end{aligned}$$

□

C.1.1 Learning Without Auxiliary OOD Data

Given in-distribution data $x_1, x_2, \ldots, x_n$, we consider the detector $G_{u,r}(x)$ with

$$u = \bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i \tag{C.9}$$

$$r = (1+\gamma/4\sqrt{d})\hat{\sigma}, \text{ where } \hat{\sigma}^2 = \frac{1}{n}\sum_{i=1}^{n} \|x_i - \bar{x}\|_2^2. \tag{C.10}$$

By concentration bounds, we can show that this leads to a good solution, if the number of data points n is sufficiently large.

Proposition 10. *If the number of in-distribution data points* $n \geqslant \frac{Cd}{\gamma^2}\log\frac{1}{\alpha} + \frac{Cd}{\gamma^4}\log\frac{1}{\alpha}$ *for* $\alpha \in (0,1)$ *and some sufficiently large constant* C*, then with probability at least* $1-\alpha$*, the detector* $G_{u,r}(x)$ *with* $u = \frac{1}{n}\sum_{i=1}^{n} x_i$ *and* $r = (1+\gamma/4\sqrt{d})\sqrt{\frac{1}{n}\sum_{i=1}^{n}\|x_i - u\|_2^2}$ *satisfies:*

$$\mathrm{FNR}(G_{u,r}) \leqslant \exp(-c\gamma^2), \tag{C.11}$$

$$\mathrm{FPR}(G_{u,r}; Q) \leqslant \epsilon_\tau, \tag{C.12}$$

for some absolute constant c*.*

Proof. We will prove that u is close to μ and r is close to $(1+c\gamma/\sqrt{d})\sigma\sqrt{d}$.

First, consider $u-\mu$. Let $x_i = \mu + \sigma g_i$ where $g_i \sim \mathcal{N}(0, I)$, and $\bar{g} = \frac{1}{n}\sum_{j=1}^{n} g_j$. Then $u - \mu = \sigma \cdot \frac{1}{n}\sum_{i=1}^{n} g_i = \sigma\bar{g}$. By the concentration of sub-gaussian variables, with probability $\geqslant 1-\alpha/2$,

$$\|\bar{g}\|_2 = \left\|\frac{1}{n}\sum_{i=1}^{n} g_i\right\|_2 \leqslant c\sqrt{\frac{d}{n}\log\frac{1}{\alpha}} \tag{C.13}$$

and thus

$$\|u-\mu\|_2 \leqslant c\sigma\sqrt{\frac{d}{n}\log\frac{1}{\alpha}} \leqslant c\sigma\gamma. \tag{C.14}$$

Next, let $\hat{\sigma}^2 := \frac{1}{n}\sum_{i=1}^{n} \|x_i - u\|_2^2$. Then

$$\hat{\sigma}^2 = \frac{\sigma^2}{n}\sum_{i=1}^{n} \|g_i - \bar{g}\|_2^2 = \sigma^2\left(\frac{1}{n}\sum_{i=1}^{n} \|g_i\|_2^2 - \sigma^2\|\bar{g}\|_2^2\right). \tag{C.15}$$

By sub-exponential concentration, we have with probability $\geqslant 1 - \alpha/2$,

$$\left|\frac{1}{n}\sum_{i=1}^{n} \|g_i\|_2^2 - d\right| \leqslant c\sqrt{\frac{d}{n}\log\frac{1}{\alpha}}. \tag{C.16}$$

Then

$$|\hat{\sigma}^2 - \sigma^2 d| \leqslant c\sigma^2\sqrt{\frac{d}{n}\log\frac{1}{\alpha}} \leqslant c\sigma^2\gamma^2. \tag{C.17}$$

Given

$$\|u - \mu\|_2 \leqslant c\sigma\gamma, |\hat{\sigma} - \sigma\sqrt{d}| \leqslant \sqrt{|\hat{\sigma}^2 - \sigma^2 d|} \leqslant c\sigma\gamma, \tag{C.18}$$

for sufficiently small $c < 1/16$, we have $(1 + \gamma/8\sqrt{d})\sigma\sqrt{d} \leqslant r \leqslant (1 + \gamma/2\sqrt{d})\sigma\sqrt{d}$. Then the statement follows. □

C.1.2 Learning With Informative Auxiliary OOD Data

The informative auxiliary data will be a distribution around the boundary of the in-distribution and the outlier distributions. Assume we have access to auxiliary OOD data x from an ideal distribution U_X where:

- U_X is a distribution over the sphere $\{x : \|x - \mu\|_2^2 = \sigma_o^2 d\}$ for some $\sigma_o > \sigma$, and its density is at least η times that of the uniform distribution on the sphere for some constant $\eta > 0$.

Given in-distribution data $x_1, \ldots, x_n$ from P_X and auxiliary OOD data $\tilde{x}_1, \ldots, \tilde{x}_{n'}$ from U_X, we now design a good detector which only requires a small number n of in-distribution data. The radius r can be estimated using a small number of

in-distribution data as above. The challenge is to learn a good u, ideally close to μ. Given an outlier data point $\tilde{x}$, a natural idea frequently used in practice is then to find a u so that not so many outliers (potentially with adversarial perturbations) can be close to u, so that $G_{u,r}$ will be able to detect the outliers. Furthermore, we know that the μ is not far from $\bar{x}$, so we only need to search u near $\bar{x}$. We thus come to the following learning rule:

$$\hat{\sigma}^2 = \frac{1}{n}\sum_{i=1}^{n} \|x_i - \bar{x}\|_2^2, \text{ where } \bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i, \tag{C.19}$$

$$r = (1 + \gamma/4\sqrt{d})\hat{\sigma}, \tag{C.20}$$

$$u \leftarrow \underset{p:\|p-\bar{x}\|_2 \leqslant s}{\arg\min}\ L_t(p) := \frac{1}{n'}\sum_{i=1}^{n'} \max_{\|\delta\|_\infty \leqslant \epsilon} \mathbb{I}\left[\|\tilde{x}_i + \delta - p\|_2 < t\right]. \tag{C.21}$$

where s and t are some hyper-parameters to be determined.

Lemma C.1. *Suppose* $\sigma^2\gamma^2 > C\epsilon\sigma_o d$, *and the number of in-distribution data points* $n \geqslant \frac{Cd}{\gamma^4}\log\frac{1}{\alpha} + \frac{C\sigma^2 d}{s^2}\log\frac{1}{\alpha}$ *and the number of auxiliary data* $n' \geqslant \frac{\exp(Cs^2/\sigma_o^2)}{\eta}\log\frac{d\sigma}{\alpha}$ *for* $\alpha \in (0,1)$ *and for some sufficiently large constant* C, *then there exists proper parameter values* t *such that with probability at least* $1-\alpha$, *the detector* $G_{u,r}(x)$ *with* r *from* (C.20) *and* u *from* (C.21) *satisfies:*

$$\mathrm{FNR}(G_{u,r}(x)) \leqslant \exp(-c\gamma^2), \tag{C.22}$$

$$\mathrm{FPR}(G_{u,r}; \mathcal{Q}) \leqslant \epsilon_\tau, \tag{C.23}$$

for some absolute constant c.

Proof. Set $B = \sigma\gamma/4$, and set t to be any value such that $\sigma_o^2 d - B^2 \leqslant t^2$ and $(t + \epsilon\sqrt{d})^2 < \sigma_o^2 d$. (Note that $\sigma_o^2 d > B^2$, since $\sigma_o^2 d > \sigma^2 d$ and $\gamma < \sqrt{d}$. Furthermore, when $\sigma^2\gamma^2 < C\epsilon\sigma_o d$ for some sufficiently constant C, we have $\sqrt{\sigma_o^2 d - B^2} < \sigma_o\sqrt{d} - \epsilon\sqrt{d}$, so we can select t within this range which will satisfy our requirements for t.)

First, consider r and $\bar{x}$. Similar to the proof of Proposition 10, when $n \geqslant \frac{cd}{\gamma^4} \log \frac{1}{\alpha}$, we have with probability $1 - \alpha/8$,

$$(1 + \gamma/8\sqrt{d})\sigma\sqrt{d} \leqslant r \leqslant (1 + \gamma/2\sqrt{d})\sigma\sqrt{d}. \tag{C.24}$$

Also for $\bar{x}$, when $n \geqslant \frac{c\sigma^2 d}{s^2} \log \frac{1}{\alpha}$, we have with probability $1 - \alpha/8$,

$$\|\mu - \bar{x}\|_2 \leqslant c\sigma\sqrt{\frac{d}{n} \log \frac{1}{\alpha}} \leqslant s. \tag{C.25}$$

This ensures μ will be included in the feasible set of (C.21).

Now, consider u. Let P be the set of p with $\|p - \bar{x}\|_2 \leqslant s$ but $\|p - \mu\|_2 > B$:

$$P := \{p \in \mathbb{R}^d : \|p - \bar{x}\|_2 \leqslant s, \|p - \mu\|_2 > B\}. \tag{C.26}$$

We will show that for any $p \in P$, with probability $1 - \alpha/8$, $L_t(p) > L_t(\mu) = 0$, and thus (C.21) finds an u with $\|u - \mu\|_2 \leqslant \sigma\gamma/4$, which then leads to the theorem statements.

We begin by noting that

$$\max_{\|\delta\|_\infty \leqslant \epsilon} \mathbb{I}\left[\|\tilde{x}_i + \delta - p\|_2 < t\right] \leqslant \mathbb{I}\left[\|\tilde{x}_i - p\|_2 < t + \epsilon\sqrt{d}\right]. \tag{C.27}$$

Since $t + \epsilon\sqrt{d} < \sigma_o\sqrt{d}$, we have $L_t(\mu) = 0$.

We now consider a fixed $p \in P$.

$$\mathbb{E}\left\{\max_{\|\delta\|_\infty \leqslant \epsilon} \mathbb{I}\left[\|\tilde{x}_i + \delta - p\|_2 < t\right]\right\} \geqslant \mathbb{E}\{\mathbb{I}\left[\|\tilde{x}_i - p\|_2 < t\right]\} = \Pr\left[\|p - \tilde{x}_i\|_2 < t\right]. \tag{C.28}$$

Let $\Delta = p - \mu$. Then since $\sigma_o^2 d - B^2 \leqslant t^2$ and $\|\Delta\|_2 \geqslant B$, we have $\sigma_o^2 d - \|\Delta\|_2^2 \leqslant t^2$, and thus any $\tilde{x}_i$ from U_X whose projection onto the direction $p - \mu$ has distance

larger than $\|\Delta\|_2$ from μ will satisfy $\|p - \tilde{x}_i\|_2 < t$. Then

$$\mathbb{E}\left\{\max_{\|\delta\|_\infty \leqslant \epsilon} \mathbb{I}\left[\|\tilde{x}_i + \delta - p\|_2 < t\right]\right\} \geqslant \Pr\left[\|p - \tilde{x}_i\|_2 < t - \epsilon\sqrt{d}\right] \geqslant \frac{\eta A_d\left(1 - \|\Delta\|_2/(\sigma_o\sqrt{d})\right)}{A_d} \tag{C.29}$$

where $A_d(\nu)$ is the area of the spherical cap of the unit hypersphere in dimension d with height ν, and A_d is the area of the whole unit hypersphere.

By the bound in Becker et al. (2016), we have

$$A_d(\nu)/A_d = d^{\Theta(1)}[1 - (1-\nu)^2]^{d/2}. \tag{C.30}$$

Since $\|\Delta\|_2 \leqslant \|p - \bar{x}\|_2 + \|\bar{x} - \mu\|_2 \leqslant 2s$, we have

$$\mathbb{E}\left\{\max_{\|\delta\|_\infty \leqslant \epsilon} \mathbb{I}\left[\|\tilde{x}_i + \delta - p\|_2 < t\right]\right\} \geqslant \frac{\eta A_d\left(1 - \|\Delta\|_2/(\sigma_o\sqrt{d})\right)}{A_d} = \eta d^{\Theta(1)}\left[1 - \frac{4s^2}{\sigma_o^2 d}\right]^{d/2} := q. \tag{C.31}$$

Then when $n' \geqslant \frac{c}{q}\log\frac{1}{\alpha'}$,

$$\Pr[L_t(p) = 0] \leqslant (1-q)^{n'} \leqslant e^{-qn'} \leqslant \alpha'/8. \tag{C.32}$$

Then a net argument on P proves that for the given n', we have with probability $\geqslant 1 - \alpha/8$, any $p \in P$ has $L_t(p) > 0$. This completes the proof. □

By setting $s = \sigma\gamma^2$ and assuming $\sigma_o^2 = O(\sigma^2)$, we have the following corollary.

Proposition 11. *Suppose* $\sigma^2\gamma^2 \geqslant C\epsilon\sigma_o d$ *for some sufficiently constant* C *and* $\sigma^2 < \sigma_o^2 < C'\sigma^2$ *for some absolute constant* C'*. For* $\alpha \in (0,1)$*, if the number of in-distribution data points* $n \geqslant \frac{Cd}{\gamma^4}\log\frac{1}{\alpha}$ *and the number of auxiliary data* $n' \geqslant \frac{\exp(C\gamma^4)}{\eta}\log\frac{d\sigma}{\alpha}$*, then there exists proper parameter values* s, t *such that with probability at least* $1 - \alpha$*, the detector*

$G_{u,r}(x)$ *with* r *from* (C.20) *and* u *from* (C.21) *satisfies:*

$$\mathrm{FNR}(G_{u,r}(x)) \leqslant \exp(-c\gamma^2), \tag{C.33}$$

$$\mathrm{FPR}(G_{u,r}; Q) \leqslant \epsilon_\tau, \tag{C.34}$$

for some absolute constant c*.*

Then we can see that this can reduce the sample size n by a factor of γ^2. For example, when $\gamma = \Theta(d^{1/8})$, we only need $n = O(\sqrt{d}\log(1/\alpha))$, while in the case without auxiliary data, we need $n = O(d^{3/4}\log(1/\alpha))$.

C.1.3 Learning With Informative Outlier Mining

The above example shows the benefit of having auxiliary OOD data for training. All the auxiliary OOD data given in the example are implicitly related to the ideal parameter for the detector μ and thus are informative for learning the detector. However, this may not be the case in practice: typically only part of the auxiliary OOD data are informative, while the remaining are not very useful or even can be harmful for the learning. Here we study such an example, and shows that how outlier mining can help to identify informative data and improve the learning performance.

The practical auxiliary data can have a lot of obvious outliers not on the boundary and also quite a few in-distribution data mixed. We model the former data as $Q_q = \mathcal{N}(0, \sigma_q^2 I)$ where σ_q is large compared to $\|\mu\|_2$, σ, and σ_o. The auxiliary data distribution U_{mix} is then a mixture of U_X and Q_q where Q_q has a large weight.

- $U_{\text{mix}} = \nu U_X + (1-2\nu)Q_q + \nu P_X$ for a small $\nu \in (0, 1)$, where $Q_q = \mathcal{N}(0, \sigma_q^2 I)$ for some large σ_q.

That is, the distribution is defined by the following process: with probability ν sample the data from the informative part U_X, and with probability $1-2\nu$ sample from the uninformative part Q_q, and with probability ν sample from the in-distribution.

Then the previous simple method will not work, and a more sophisticated method is needed. Below we show that outlier mining can remove most data outside U_X, and keep the data from U_X, and the previous method can work after outlier mining.

Suppose the algorithm gets n in-distribution data $S_{in} = \{x_1, x_2, \ldots, x_n\}$ i.i.d. from P_X and n' auxiliary data $S_{au} = \{\tilde{x}_1, \tilde{x}_2, \ldots, \tilde{x}_{n'}\}$ from U_{mix} for training. Specifically, we first use in-distribution data to get an intermediate solution $\bar{x}$ and r:

$$\hat{\sigma}^2 = \frac{1}{n}\sum_{i=1}^{n} \|x_i - \bar{x}\|_2^2, \text{ where } \bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i, \tag{C.35}$$

$$r = (1 + \gamma/4\sqrt{d})\hat{\sigma}. \tag{C.36}$$

Then, we use a simple thresholding mechanism to only pick points close to the decision boundary of the intermediate solution, which removes *non-informative outliers*. Specifically, we only select outliers $\tilde{x}$ with mild "confidence scores" w.r.t. the intermediate solution, i.e., the distances to $\bar{x}$ fall in an interval $[a, b]$:

$$S := \{i : \|\tilde{x}_i - \bar{x}\|_2 \in [a, b], 1 \leqslant i \leqslant n'\} \tag{C.37}$$

The final solution u is then by:

$$u_{om} \leftarrow \operatorname*{arg\,min}_{p:\|p-\bar{x}\|_2 \leqslant s} L_{S,t}(p) := \frac{1}{|S|}\sum_{i\in S} \max_{\|\delta\|_\infty \leqslant \epsilon} \mathbb{I}\left[\, \|\tilde{x}_i + \delta - p\|_2 < t\right]. \tag{C.38}$$

We can prove the following:

Lemma C.2. *Suppose* $\sigma^2\gamma^2 \geqslant C\epsilon\sigma_o d$ *and* $\sigma_o\sqrt{d} > \sigma\sqrt{d} + Cs$ *for a sufficiently large constant* C, *and* $\sigma_q\sqrt{d} > 2(\sigma_o\sqrt{d} + \|\mu\|_2)$. *For some absolute constant* c *and any* $\alpha \in (0, 1)$, *if the number of in-distribution data* $n \geqslant \frac{Cd}{\gamma^4}\log\frac{1}{\alpha} + \frac{C\sigma^2 d}{s^2}\log\frac{1}{\alpha}$ *and the number of auxiliary data* $n' \geqslant \frac{\exp(Cs^2/\sigma_o^2)}{\nu^2\eta^2}\log\frac{d\sigma}{\alpha}$, *then there exists parameter values* t, a, b

such that with probability $\geqslant 1-\alpha$*, the detector* $G_{u_{om},r}$ *computed above satisfies:*

$$\mathrm{FNR}(G_{u_{om},r}) \leqslant \exp(-c\gamma^2), \tag{C.39}$$

$$\mathrm{FPR}(G_{u,r}; \mathcal{Q}) \leqslant \epsilon_\tau. \tag{C.40}$$

Proof. Following the proof as in Proposition 11, we have the same guarantees for r and $\bar{x}$, i.e., with probability $1-\alpha/4$,

$$(1+\gamma/8\sqrt{d})\sigma\sqrt{d} \leqslant r \leqslant (1+\gamma/2\sqrt{d})\sigma\sqrt{d}, \tag{C.41}$$

$$\|\mu - \bar{x}\|_2 \leqslant c\sigma\sqrt{\frac{d}{n}\log\frac{1}{\alpha}} \leqslant s. \tag{C.42}$$

This ensures μ will be included in the feasible set of (C.38).

Set $B = \sigma\gamma/4$, and set t to be any value such that $\sigma_o^2 d - B^2 \leqslant t^2$ and $(t+\epsilon\sqrt{d})^2 < \sigma_o^2 d$. (As before, the assumptions make sure we can select such a t. We have $\sigma_o^2 d > B^2$, since $\sigma_o^2 d > \sigma^2 d$ and $\gamma < \sqrt{d}$. Furthermore, when $\sigma^2\gamma^2 < C\epsilon\sigma_o d$ for some sufficiently constant C, we have $\sqrt{\sigma_o^2 d - B^2} < \sigma_o\sqrt{d} - \epsilon\sqrt{d}$, so we can select t within this range which will satisfy our requirements for t.)

Set $a = \sigma_o\sqrt{d} - s$, and $b = \sigma_o\sqrt{d} + s$. Let P be the set of p with $\|p-\bar{x}\|_2 \leqslant s$ but $\|p-\mu\|_2 > B$:

$$P := \{p \in \mathbb{R}^d : \|p-\bar{x}\|_2 \leqslant s, \|p-\mu\|_2 > B\}. \tag{C.43}$$

We will show that for any $p \in P$, with probability $1-\alpha/4$, $L_t(p) > L_t(\mu) = 0$, and thus (C.38) finds a u_{om} with $\|u_{om}-\mu\|_2 \leqslant B = \sigma\gamma/4$, which then leads to the theorem statements.

First, we show that $L_{S,t}(p)$ is large for any $p \in P$. The same proof as in Proposition 11 shows that for any p as described above, for an $\tilde{x}_i$ from U_X, we have

$$\mathbb{E}\left\{\max_{\|\delta\|_\infty \leqslant \epsilon} \mathbb{I}\left[\|\tilde{x}_i + \delta - p\|_2 < t\right]\right\} \geqslant \Pr\left[\|p-\tilde{x}_i\| < t\right] \geqslant \eta d^{\Theta(1)}\left[1 - \frac{4s^2}{\sigma_o^2 d}\right]^{d/2} := q. \tag{C.44}$$

We note that all points from U_X will be in S for the given a and b. Then by the Chernoff's inequality (for multiplicative factors), when $n' \geqslant \frac{c}{q\nu} \log \frac{1}{\alpha'}$, with probability $1 - \alpha'$, we have

$$L_{S,t}(p) > \frac{q\nu}{2}. \tag{C.45}$$

Then a net argument on P proves that with the given n', we have with probability $\geqslant 1 - \alpha/8$, any $p \in P$ has $L_{S,t}(p) > q\nu/2$.

Next, we will show that $L_{S,t}(\mu)$ is small. We first note that since $t + \epsilon\sqrt{d} < \sigma_o\sqrt{d}$, all points from U_X will contribute 0 to $L_{S,t}(\mu)$. Furthermore, most points from P_X and Q_q are filtered outside S.

$$\begin{aligned}
\Pr_{x\sim P_X}[\|x - \bar{x}\|_2 \in [a, b]] &\leqslant \Pr_{x\sim P_X}[\|x - \bar{x}\|_2 \geqslant a] && \text{(C.46)}\\
&\leqslant \Pr_{x\sim P_X}\left[\|x - \mu\|_2 \geqslant a - s = \sigma_o\sqrt{d} - 2s\right] && \text{(C.47)}\\
&\leqslant e^{-c(\sigma_o\sqrt{d}-2s)^2/\sigma^2} := q_1, && \text{(C.48)}
\end{aligned}$$

and

$$\begin{aligned}
\Pr_{x\sim Q_q}[\|x - \bar{x}\|_2 \in [a, b]] &\leqslant \Pr_{x\sim Q_q}[\|x - \bar{x}\|_2 \leqslant b] && \text{(C.49)}\\
&\leqslant \Pr_{x\sim Q_q}[\|x\|_2 \leqslant b + s + \|\mu\|_2] && \text{(C.50)}\\
&\leqslant e^{-c(\sigma_q\sqrt{d}-\sigma_o\sqrt{d}-2s-\|\mu\|_2)^2} := q_2. && \text{(C.51)}
\end{aligned}$$

Then by Hoeffding's inequality, when $n' \geqslant \frac{c}{q^2\nu^2} \log \frac{1}{\alpha}$, we have with probability $\geqslant 1 - \alpha/8$,

$$L_{S,t}(\mu) < q\nu/4 + \nu q_1 + (1 - \nu)q_2. \tag{C.52}$$

Note that $q \geqslant \eta d^{\Theta(1)} e^{-cs^2/\sigma_o^2}$. Then when $\sigma_o\sqrt{d} > \sigma\sqrt{d} + Cs$, we have $q_1 \leqslant e^{-cd}$. Since d is sufficiently large compared to s^2/σ_o^2, $q_1 \leqslant q/8$. Similarly, when $\sigma_q\sqrt{d} > 2(\sigma_o\sqrt{d} + \|\mu\|_2)$, we have $\sigma_q\sqrt{d} - \sigma_o\sqrt{d} - 2s - \|\mu\|_2 > \sigma_q\sqrt{d}/4$, and thus $q_2 \leqslant \nu q/8$.

Therefore, $L_{S,t}(\mu) < q\nu/2$.

In summary, with probability $\geqslant 1-\alpha$, $L_{S,t}(\mu) < L_{S,t}(p)$ for any $p \in P$. This then completes the proof. □

By setting $s = \sigma\gamma^2$ and assuming $\sigma_o^2 = O(\sigma^2)$, we have the following corollary.

Proposition 12 (Restatement of Proposition 5)**.** (***Error bound with outlier mining.***) *Suppose* $\sigma^2\gamma^2 \geqslant C\epsilon\sigma_o d$ *and* $\sigma\sqrt{d} + C\sigma\gamma^2 < \sigma_o\sqrt{d} < C\sigma\sqrt{d}$ *for a sufficiently large constant* C*, and* $\sigma_q\sqrt{d} > 2(\sigma_o\sqrt{d} + \|\mu\|_2)$*. For any* $\alpha \in (0,1)$ *and some absolute constant* c*, if the number of in-distribution data* $n \geqslant \frac{Cd}{\gamma^4}\log\frac{1}{\alpha}$ *and the number of auxiliary data* $n' \geqslant \frac{\exp(C\gamma^4)}{\nu^2\eta^2}\log\frac{d\sigma}{\alpha}$*, then there exist parameter values* s, t, a, b *such that with probability* $\geqslant 1-\alpha$*, the detector* $G_{u_{om},r}$ *computed above satisfies:*

$$\begin{aligned} \mathrm{FNR}(G_{u_{om},r}) &\leqslant \exp(-c\gamma^2), \\ \mathrm{FPR}(G_{u_{om},r}; Q) &\leqslant \epsilon_\tau. \end{aligned}$$

This means that even in the presence of a large amount of uninformative or even harmful auxiliary data, we can successfully learn a good detector. Furthermore, this can reduce the sample size n by a factor of γ^2. For example, when $\gamma = \Theta(d^{1/8})$, we only need $n = O(\sqrt{d}\log(1/\alpha))$, while in the case without auxiliary data, we need $n = O(d^{3/4}\log(1/\alpha))$.

C.1.4 Learning with Ideal U_X

Consider the same setting in the above Section C.1.3 but assume that the U_X in the mixture U_{mix} is as ideal as a uniform distribution:

- U_X is the uniform distribution over the sphere $\{x : \|x-\mu\|_2^2 = \sigma_o^2 d\}$ where $\sigma_o^2 > \sigma^2$.

In this case we can do the outlier mining as above, but finally compute u_{om} by

averaging the auxiliary data points selected in S. That is:

$$\hat{\sigma}^2 = \frac{1}{n}\sum_{i=1}^{n} \|x_i - \bar{x}\|_2^2, \text{ where } \bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i, \tag{C.53}$$

$$r = (1 + \gamma/4\sqrt{d})\hat{\sigma}, \tag{C.54}$$

$$S = \{i : \|\tilde{x}_i - \bar{x}\|_2 \in [a, b], 1 \leqslant i \leqslant n'\}, \tag{C.55}$$

$$u_{om} = \frac{1}{|S|}\sum_{\bar{x}\in S} \bar{x}. \tag{C.56}$$

Then we can prove essentially the same guarantees but with much fewer auxiliary data:

Proposition 13. *Suppose* $\sigma^2\gamma^2 \geqslant C\epsilon\sigma_o d$ *and* $\sigma\sqrt{d} + C\sigma\gamma^2 < \sigma_o\sqrt{d} < C\sigma\sqrt{d}$ *for a sufficiently large constant* C*, and* $\sigma_q\sqrt{d} > 2(\sigma_o\sqrt{d} + \|\mu\|_2)$*. For any* $\alpha \in (0, 1)$ *and some absolute constant* c*, if the number of in-distribution data* $n \geqslant \frac{Cd}{\gamma^4}\log\frac{1}{\alpha}$ *and the number of auxiliary data* $n' \geqslant \frac{Cd}{\gamma^2\gamma^2}\log\frac{d}{\alpha}$ *and* $d \geqslant C\log\frac{n'}{\alpha}$*, then there exist parameter values* a, b *such that with probability* $\geqslant 1 - \alpha$*, the detector* $G_{u_{om},r}$ *computed above satisfies:*

$$\begin{aligned} \mathrm{FNR}(G_{u_{om},r}) &\leqslant \exp(-c\gamma^2), \\ \mathrm{FPR}(G_{u_{om},r}; Q) &\leqslant \epsilon_\tau. \end{aligned}$$

Proof. Following the same setting of a, b and the same proof as in Lemma C.2, we can show that all points from U_X will be included in S, and most points from P_X and Q_q are filtered outside S:

$$\Pr_{x\sim P_X}[\|x - \bar{x}\|_2 \in [a, b]] \leqslant e^{-c(\sigma_o\sqrt{d}-2s)^2/\sigma^2} < e^{-c'd}, \tag{C.57}$$

$$\Pr_{x\sim Q_q}[\|x - \bar{x}\|_2 \in [a, b]] \leqslant e^{-c(\sigma_q\sqrt{d}-\sigma_o\sqrt{d}-2s-\|\mu\|_2)^2} < e^{-c'd}, \tag{C.58}$$

for some absolute constant c'. Then with probability $\geqslant 1 - n'e^{-2c'd}$, S is exactly all the auxiliary points from U_X. By the sub-gaussian concentration of the uniform

distribution over sphere, we have with probability $1 - \alpha/2$,

$$\|u_{om} - \mu\|_2 \leqslant c\sigma_o \sqrt{\frac{d}{|S|} \log \frac{1}{\alpha}}. \tag{C.59}$$

For the given π', we have the theorem. □

C.2 Adversarial Attacks for OOD Detection Methods

We propose adversarial attack objectives for different OOD detection methods. We consider a family of adversarial perturbations for the OOD inputs: (1) L_∞-norm bounded attack (white-box attack); (2) common image corruptions attack (black-box attack); (3) compositional attack which combines common image corruptions attack and L_∞ norm bounded attack (white-box attack).

L_∞ norm bounded attack. For data point $x \in \mathbb{R}^d$, the L_∞ norm bounded perturbation is defined as

$$\Omega_{\infty,\epsilon}(x) = \{\delta \in \mathbb{R}^d \mid \|\delta\|_\infty \leqslant \epsilon \wedge x + \delta \text{ is valid}\}, \tag{C.60}$$

where ϵ is the adversarial budget. $x + \delta$ is considered valid if the values of $x + \delta$ are in the image pixel value range.

For MSP, ODIN, OE, ACET, and CCU methods, we propose the following attack objective to generate adversarial OOD example on a clean OOD input x:

$$x' = \underset{x' \in \Omega_{\infty,\epsilon}(x)}{\arg\max} -\frac{1}{K} \sum_{i=1}^{K} \log F(x')_i \tag{C.61}$$

where $F(x)$ is the softmax output of the classifier network.

For Mahalanobis method, we propose the following attack objective to generate

adverasrial OOD example on OOD input x:

$$x' = \underset{x' \in \Omega_{\infty,\epsilon}(x)}{\arg\max} \; -\log \frac{1}{1 + e^{-(\sum_\ell \alpha_\ell M_\ell(x') + b)}}, \tag{C.62}$$

where $M_\ell(x')$ is the Mahalanobis distance-based confidence score of x' from the ℓ-th feature layer, $\{\alpha_\ell\}$ and b are the parameters of the logistic regression model.

For SOFL method, we propose the following attack objective to generate adversarial OOD example for an input x:

$$x' = \underset{x' \in \Omega_{\infty,\epsilon}(x)}{\arg\max} \; -\log \sum_{i=K+1}^{K+R} \bar{F}(x')_i \tag{C.63}$$

where $\bar{F}(x)$ is the softmax output of the whole neural network (including auxiliary head) and R is the number of reject classes.

For ROWL and ATOM method, we propose the following attack objective to generate adverasrial OOD example on OOD input x:

$$x' = \underset{x' \in \Omega_{\infty,\epsilon}(x)}{\arg\max} \; -\log \hat{F}(x')_{K+1} \tag{C.64}$$

where $\hat{F}(x)$ is the softmax output of the (K+1)-way neural network.

Due to computational concerns, by default, we will use PGD with $\epsilon = 8/255$, the number of iterations of 40, the step size of $1/255$ and random start to solve these attack objectives.

Common Image Corruptions attack. We use common image corruptions introduced in Hendrycks and Dietterich (2019). We apply 15 types of algorithmically generated corruptions from noise, blur, weather, and digital categories to each OOD image. Each type of corruption has five levels of severity, resulting in 75 distinct corruptions. Thus, for each OOD image, we generate 75 corrupted images and then select the one with the lowest OOD score (or highest confidence score to be in-distribution). Note that we only need the outputs of the OOD detectors to

construct such adversarial OOD examples; thus it is a black-box attack.

Compositional Attack. For each OOD image, we first apply common image corruptions attack, and then apply the L_∞-norm bounded attack to generate adversarial OOD examples.

D.1 Complete Proofs and Discussions

We first present the proof for our main theorem, and then provide some more discussion on the benefit of using ensembles.

D.1.1 Provable Guarantees of the Framework

We first prove a technical lemma for constructing R and then use it to prove the theorem.

First recall the key notions.

Suppose on points where f is correct, the ensemble models are also approximately correct. Let ν denote the average probability of $h \sim \mathcal{T}$ making error on the test points where the pre-trained model f is correct:

$$\nu := \Pr_{(x,y_x)\sim U, h\sim\mathcal{T}}[h(x) \neq y_x | f(x) = y_x]. \tag{D.1}$$

Suppose the ensemble training with regularization on the pseudo-labeled data R will make the ensemble models disagree with f on R_X. Let γ denote the average probability of $h \sim \mathcal{T}$ agreeing with f on the test points in R_X:

$$\gamma := \Pr_{x\sim R_X, h\sim\mathcal{T}}\{h(x) = f(x)\}. \tag{D.2}$$

Suppose G_X is the points in $W_X \setminus R_X$ on which the ensemble agree with the true label y_x with more than $1-\nu$ probability, i.e.,

$$G_X := \{x \in W_X \setminus R_X : \Pr_{h\sim\mathcal{T}}\{h(x) = y_x\} \geqslant 1-\nu\}. \tag{D.3}$$

That is, G_X are the points where the ensemble will have correct prediction with high confidence. We would like the ensemble to have large diversity on the remaining

points in $W_X \setminus R_X$. Define the diversity there to be

$$B_X := W_X \setminus (R_X \cup G_X), \tag{D.4}$$

$$\sigma^2 := \mathbb{E}[\sigma_x^2 | x \in B_X]. \tag{D.5}$$

Lemma D.1. *Define*

$$B_\eta := \min\{\sigma^2, 1 - \nu^2\}. \tag{D.6}$$

For any $\eta \in (0, B_\eta)$, *let* $\tau = \sqrt{1-\eta}$, *and*

$$R'_X := \left\{ x \in U_X : \Pr_{h \sim \mathcal{T}}\{h(x) = f(x)\} < \tau \right\}. \tag{D.7}$$

Then we have

$$|R'_X \cap (U_X \setminus W_X)| \leqslant \frac{\nu}{1-\tau} |U_X \setminus W_X|, \text{ and}$$

$$(1 - \frac{\gamma}{\tau})|R_X| \leqslant |R_X \cap R'_X|, G_X \subseteq R'_X, |R'_X \cap B_X| \geqslant \frac{\sigma^2 - \eta}{1-\eta} |B_X|.$$

Proof. Consider $x \in U_X - W_X$. We have

$$\Pr_{(x, y_x) \sim U} \left\{ \Pr_{h \sim \mathcal{T}}\{h(x) = f(x)\} < \tau | f(x) = y_x \right\} \tag{D.8}$$

$$= \Pr_{(x, y_x) \sim U} \left\{ \Pr_{h \sim \mathcal{T}}\{h(x) \neq y_x\} \geqslant 1 - \tau | f(x) = y_x \right\} \tag{D.9}$$

$$\leqslant \frac{\nu}{1-\tau}. \tag{D.10}$$

So only $\frac{\nu}{1-\tau}$ fraction of the data points in $U_X \setminus W_X$ will be put into R_X, proving the first statement.

Now, consider $x \in W_X$. For $x \in R_X$, we have

$$\Pr_{x \sim R_X} \left\{ \Pr_{h \sim \mathcal{T}}\{h(x) = f(x)\} \geqslant \tau \right\} \leqslant \frac{\gamma}{\tau}. \tag{D.11}$$

so more than $1 - \frac{\gamma}{\tau}$ fraction of the data points in R_X will be put into R'_X.

For $x \in G_X$, since $\eta < 1 - \nu^2$, we have $\nu < \tau$. Note that $f(x) \neq y_x$, thus x will be put into R'_X. In the following we consider $x \in B_X = W_X \setminus (R_X \cup G_X)$.

We first show that a significant fraction of B_X has variance larger than η. Since $\sigma_x^2 \in [0, 1]$,

$$\sigma^2 = \mathbb{E}[\sigma_x^2 | x \in B_X] \tag{D.12}$$

$$\leqslant \Pr\{\sigma_x^2 \leqslant \eta | x \in B_X\} \cdot \eta + \Pr\{\sigma_x^2 > \eta | x \in B_X\} \cdot 1 \tag{D.13}$$

$$= (1 - \Pr\{\sigma_x^2 > \eta | x \in B_X\}) \cdot \eta + \Pr\{\sigma_x^2 > \eta | x \in B_X\} \tag{D.14}$$

leading to

$$\Pr\{\sigma_x^2 > \eta | x \in B_X\} \geqslant \frac{\sigma^2 - \eta}{1 - \eta}. \tag{D.15}$$

Now it is sufficient to show that any $x \in W_X$ with $\sigma_x^2 > \eta$ will have a small agreement rate $\Pr_{h \sim \mathcal{T}}\{h(x) = f(x)\}$.

$$\Pr_{h \sim \mathcal{T}}[h(x) = f(x)] \leqslant \sqrt{\sum_{y \in \mathcal{Y}} (\Pr_{h \sim \mathcal{T}}[h(x) = y])^2} \tag{D.16}$$

$$= \sqrt{\Pr_{h_1, h_2 \sim \mathcal{T}}[h_1(x) = h_2(x)]} \tag{D.17}$$

$$= \sqrt{1 - \sigma_x^2} \tag{D.18}$$

$$< \tau = \sqrt{1 - \eta}. \tag{D.19}$$

So any point $x \in B_X$ with $\sigma_x^2 > \eta$ will fall into R'_X, completing the proof. □

Using the above lemma, we arrive at our main theorem.

Theorem D.2 (Restatement of Theorem 6.4)**.** *Assume in each iteration of the framework, $\tau = \sqrt{1 - \eta}$ for some $\eta \in (0, 3B_\eta/4)$ where $B_\eta := \min\{\sigma^2, 1 - \nu^2\}$. Let $\sigma_L^2 > 0$ be a lower bound on the diversity σ^2, $\tilde{\gamma}$ be an upper bound on γ, and $\tilde{\nu}$ be an upper bound on ν over all iterations. Then for any $\delta \in (0, \sigma_L^2/4)$, after at most $\lceil 1/\delta \rceil$ iterations, we can get $\frac{|U_X \setminus R_X|}{|U_X|}$*

approximates the accuracy $\mathrm{acc}(f)$ *and* R_X *approximates the mis-classified points* W_X *as follows:*

$$\left|\mathrm{acc}(f) - \frac{|U_X \setminus R_X|}{|U_X|}\right| \leqslant \max\{\frac{\tilde{\nu}}{1-\tau}(1-e_f), \epsilon e_f\}, \text{ where } \epsilon := \frac{\frac{\tilde{\gamma}}{\tau}\left(1 + \frac{\tilde{\nu}}{1-\tau}\frac{1-e_f}{e_f}\right)}{\frac{\sigma_L^2}{4} - \delta + \frac{\tilde{\gamma}}{\tau}}, \tag{D.20}$$

$$|W_X \triangle R_X| \leqslant \frac{\tilde{\nu}}{1-\tau}|U_X \setminus W_X| + \epsilon|W_X|. \tag{D.21}$$

Proof. For each iteration, Lemma D.1 implies that the new constructed set R'_X satisfies:

$$|R'_X \cap (U_X \setminus W_X)| \leqslant \frac{\nu}{1-\tau}|U_X \setminus W_X|, \text{ and}$$
$$(1 - \frac{\gamma}{\tau})|R_X| \leqslant |R_X \cap R'_X|, G_X \subseteq R'_X, |R'_X \cap B_X| \geqslant \frac{\sigma^2 - \eta}{1-\eta}|B_X|.$$

Since $\eta \leqslant 3\sigma^2/4$,

$$\frac{\sigma^2 - \eta}{1-\eta} \geqslant \frac{\sigma^2}{4}. \tag{D.22}$$

Therefore,

$$|R'_X \cap B_X| \geqslant \frac{\sigma^2}{4}|B_X|. \tag{D.23}$$

Suppose t^* is the first iteration when less than ϵ fraction of W_X is outside R_X. Then in any iteration before t^*, the newly constructed pseudo-labeled set R'_X loses at most $\frac{\tilde{\gamma}}{\tau}$ fraction of R_X, and obtains at least $\frac{\sigma_L^2}{4}$ fraction of $B_X \cap G_X = W_X \setminus R_X$. Since more than ϵ fraction of W_X is outside R_X, it can then be verified that

$$\frac{\sigma_L^2}{4}|W_X \setminus R_X| - \frac{\tilde{\gamma}}{\tau}|R_X| > \delta|W_X \setminus R_X|. \tag{D.24}$$

Therefore, after each iteration, the framework adds more than δ fraction of $|W_X \setminus R_X|$

in R_X. This can happen at most $1/\delta$ iterations, so $t^* \leqslant \lceil 1/\delta \rceil$.

Now consider the last iteration, and apply Lemma D.1, then

$$|R_X \setminus W_X| = |R_X \cap (U_X \setminus W_X)| \leqslant \frac{\nu}{1-\tau}|U_X \setminus W_X| \leqslant \frac{\tilde{\nu}}{1-\tau}|U_X \setminus W_X|. \tag{D.25}$$

Equation (D.25) together with the fact that there are less than ϵ fraction of W_X outside R_X lead to the two statements in the theorem. □

By setting the $\eta = 7/16$ and $\delta = 4\tilde{\gamma}/3$, we have the following corollary as an example.

Corollary D.3. *Assume in each iteration of the framework, $\nu < 1/2, \sigma^2 > 7/12$, and $\tau = 3/4$ where $B_\eta := \min\{\sigma^2, 1-\nu^2\}$. Let $\sigma_L^2 > 0$ be a lower bound on the diversity σ^2, $\tilde{\gamma}$ be an upper bound on γ, and $\tilde{\nu}$ be an upper bound on ν over all iterations. If $\sigma_L^2 \geqslant \frac{16\tilde{\gamma}}{3}$, then after at most $\lceil 3/(4\tilde{\gamma}) \rceil$ iterations, we can get $\frac{|U_X \setminus R_X|}{|U_X|}$ approximates the accuracy $\mathrm{acc}(f)$ and R_X approximates the mis-classified points W_X as follows:*

$$\left| \mathrm{acc}(f) - \frac{|U_X \setminus R_X|}{|U_X|} \right| \leqslant \max\{4\tilde{\nu}(1-e_f), \epsilon e_f\}, \text{ where } \epsilon := \frac{16\tilde{\gamma}}{3\sigma_L^2}\left(1 + 4\tilde{\nu}\frac{1-e_f}{e_f}\right), \tag{D.26}$$

$$|W_X \triangle R_X| \leqslant 4\tilde{\nu}|U_X \setminus W_X| + \epsilon|W_X|. \tag{D.27}$$

Proof. In Theorem 6.4, note that $\eta = 7/16$ leads to $\tau = 3/4$. The bounds on ν, γ, and σ^2 comes from the requirement that $3B_\eta/4 > 7/16$ so that there exists such an $\eta \in (7/16, 3B_\eta/4)$. □

D.1.2 Discussion on Using Ensembles

Our analysis of the framework clearly relies on the effect of self-training. It also shows the benefit of the ensemble: the diversity (combined with low errors of the ensemble on points correctly classified by f) allows to identify mis-classified points.

Here we present more discussion on the approach of using ensembles to estimate the accuracy and to provide further insight into their benefit compared to some other

existing approaches. For simplicity, we analyze binary classification with $\mathcal{Y} = \{0, 1\}$ in this section unless stated otherwise. We provide an exact characterization of the estimation error (i.e., how far the agreement rate is from the actual accuracy). It implies that to get a good estimation, one should use ensembles with small prediction errors on the test points. More importantly, the estimation can be further improved if the ensemble's prediction has proper correlation with f, which then shows the advantage of an ensemble of models instead of a single model.

Let $e_{h,x}$ be the indicator that h mis-classifies x, $e_{\mathcal{T}}$ be the expected error of h on U (over the distribution of h), and e_f be the error of f:

$$e_{h,x} := \mathbb{I}[h(x) \neq y_x], \quad e_{\mathcal{T}} := \mathbb{E}_{h,x}[e_{h,x}], \quad e_f := \mathbb{E}_x[e_{f,x}].$$

Let $\mathrm{ar}_x(f, \mathcal{T})$ be the agreement rate between f and h's on a point x, and $\mathrm{ar}(f, \mathcal{T})$ be that on the whole test set:

$$\mathrm{ar}_x(f, \mathcal{T}) := \Pr_{h \sim \mathcal{T}}\{h(x) = f(x)\},$$
$$\mathrm{ar}(f, \mathcal{T}) := \mathbb{E}_{x \in U_X}[\mathrm{ar}_x(f, \mathcal{T})].$$

Recall that we are using $\mathrm{ar}(f, \mathcal{T})$ as an estimate of the accuracy of f on U.

Lemma D.4. *For binary classification,*

$$\mathrm{acc}(f) - \mathrm{ar}(f, \mathcal{T}) = e_{\mathcal{T}}(1 - 2e_f) - 2\mathrm{Cov}(e_{f,x}, e_{h,x}) \tag{D.28}$$

where $\mathrm{Cov}(e_{f,x}, e_{h,x})$ *is the covariance between* $e_{f,x}$ *and* $e_{h,x}$*. For multi-class classification,*

$$e_{\mathcal{T}}(1 - 2e_f) - 2\mathrm{Cov}(e_{f,x}, e_{h,x}) \leqslant \mathrm{acc}(f) - \mathrm{ar}(f, \mathcal{T}) \leqslant e_{\mathcal{T}}(1 - e_f) - \mathrm{Cov}(e_{f,x}, e_{h,x}). \tag{D.29}$$

Proof. We have

$$\begin{aligned}
\mathrm{acc}(f) - \mathrm{ar}(f, \mathcal{T}) &= \mathbb{E}\{\mathbb{I}[f(x) = y_x]\} - \mathbb{E}\{\mathbb{I}[f(x) = h(x)]\} && \text{(D.30)}\\
&= \mathbb{E}\{\mathbb{I}[f(x) = y_x] - \mathbb{I}[f(x) = h(x)]\} && \text{(D.31)}\\
&= \mathbb{E}\{\mathbb{I}[f(x) \neq h(x)] - \mathbb{I}[f(x) \neq y_x]\}. && \text{(D.32)}
\end{aligned}$$

The first term can be decomposed into two parts:

$$\mathbb{I}[f(x) \neq h(x)] = \mathbb{I}[f(x) \neq h(x), f(x) = y_x] + \mathbb{I}[f(x) \neq h(x), f(x) \neq y_x] \quad \text{(D.33)}$$

and the two parts can be transformed as:

$$\begin{aligned}
\mathbb{I}[f(x) \neq h(x), f(x) = y_x] &= \mathbb{I}[h(x) \neq y_x, f(x) = y_x] && \text{(D.34)}\\
&= e_{h,x}(1 - e_{f,x}), && \text{(D.35)}\\
\mathbb{I}[f(x) \neq h(x), f(x) \neq y_x] &= \mathbb{I}[h(x) = y_x, f(x) \neq y_x] && \text{(D.36)}\\
&= e_{f,x}(1 - e_{h,x}) && \text{(D.37)}
\end{aligned}$$

where the second to last line follows from that in binary classification, $f(x) \neq h(x)$ and $f(x) \neq y_x$ is equivalent to $h(x) = y_x$ and $f(x) \neq y_x$. Therefore,

$$\begin{aligned}
\mathrm{acc}(f) - \mathrm{ar}(f, \mathcal{T}) &= e_{\mathcal{T}} - 2\mathbb{E}[e_{h,x} e_{f,x}] && \text{(D.38)}\\
&= e_{\mathcal{T}} - 2(e_{\mathcal{T}} e_f + \mathrm{Cov}(e_{h,x}, e_{f,x})). && \text{(D.39)}
\end{aligned}$$

Rearranging the terms completes the proof.

For multi-class, we can replace (D.36) by the bounds:

$$\mathbb{I}[h(x) = y_x, f(x) \neq y_x] \leqslant \mathbb{I}[f(x) \neq h(x), f(x) \neq y_x] \leqslant \mathbb{I}[f(x) \neq y_x]. \quad \text{(D.40)}$$

□

The bound suggests using $\mathcal{T}$ with a small prediction error $e_{\mathcal{T}}$. More importantly, the estimation can be improved by a proper correlation between f and the ensemble

models: even when the ensemble models don't have very small error $e_{\mathcal{T}}$, they can still lead to a good estimation, as long as they have a proper covariance with f. More precisely, typically $e_f < 1/2$, so the covariance should not be negative, but also should not be too positive. For example, when the ensemble models overly agrees with f (e.g., in the extreme case $h(x) = f(x)$ for all $x \in U_X$ and all $h \sim \mathcal{T}$), it leads to over-estimation of the accuracy, and we should decrease the correlation (more discussion in the next subsection).

It is also instructive to compare our method to some existing methods. (1) Our analysis is more general and tighter than that for using a single model in Chuang et al. (2020). The setting is a special case of ours. More important, our bound is tighter and reveals that an ensemble with proper correlation can improve the estimation, justifying the advantage of an ensemble over a single model. (2) Our analysis is also more general than the classic notion of calibration. We show that if the ensemble has perfect calibration then the agreement rate equals the accuracy of the pre-trained model. On the other hand, our lemma shows that even without calibration, proper ensembles can still give good estimation.

Detailed comparisons are presented below.

Comparison with Proxy Risk. Recall that the proxy risk method (Chuang et al., 2020) is to use invariant representation domain adaptation methods to find the h of maximum disagreement with f on U_X. That is, it aims to get the $h \in \mathcal{H}$ such that $\text{ar}(f, h)$ is smallest where $\mathcal{H}$ is the set of hypotheses with small errors on the original training data and small distances between the distributions of the representations of the training and test data, i.e.,

$$\mathcal{H} = \{h \in \mathcal{P} : \text{ error of } h \text{ on the training data } + \alpha d(p_S^\phi, p_T^\phi) \leqslant \epsilon\} \tag{D.41}$$

where $\mathcal{P}$ is the set of networks for domain adaptation, $d(p_S^\phi, p_T^\phi)$ is some distance between the distributions of the representations of the training and the test data, and α, ϵ are hyperparameters.

The main idea behind the proxy risk method is Lemma 4 in their paper, which

states (rephrased to our context):

$$\left|\sup_{h\in\mathcal{H}} \mathbb{E}_{x\sim U_X}\mathbb{I}[f(x)\neq h(x)] - e_f\right| \leqslant \sup_{h\in\mathcal{H}} e_h \tag{D.42}$$

where e_h is the error of h on the test set, i.e., $e_h = \mathbb{E}_{(x,y_x)\sim U}\{\mathbb{I}[h(x)\neq y_x]\}$.

Our bound is more general and tighter. We first show that their bound can be recovered from ours. More precisely, the proxy risk method is equivalent to using an ensemble method $\mathcal{T}$ that outputs the $\hat{h}\in\mathcal{H}$ of maximum disagreement with f. Then the output distribution of $\mathcal{T}$ concentrates on $\hat{h}\in\mathcal{H}$. Our bound then leads to:

$$\left|\sup_{h\in\mathcal{H}} \mathbb{E}_x\mathbb{I}[f(x)\neq h(x)] - e_f\right| = \left|\mathbb{E}_x\mathbb{I}[f(x)\neq \hat{h}(x)] - e_f\right| \tag{D.43}$$

$$= \left|\mathrm{acc}(f) - \mathbb{E}_{h\sim\mathcal{T}}[\mathrm{ar}(f,h)]\right| \tag{D.44}$$

$$= |e_{\mathcal{T}} - 2\mathbb{E}[e_{f,x}e_{h,x}]| \tag{D.45}$$

$$= |e_{\hat{h}} - 2\mathbb{E}_x[e_{f,x}e_{h,x}]|. \tag{D.46}$$

Since $e_{f,x}$ and $e_{h,x}$ are in $\{0,1\}$, it is easy to see that

$$0 \leqslant \mathbb{E}_x[e_{f,x}e_{h,x}] \leqslant \min\{\mathbb{E}_x[e_{f,x}], \mathbb{E}_x[e_{\hat{h},x}]\} = \min\{e_f, e_{\hat{h}}\}. \tag{D.47}$$

Therefore,

$$\left|\sup_{h\in\mathcal{H}} \mathbb{E}_x\mathbb{I}[f(x)\neq h(x)] - e_f\right| = |e_{\hat{h}} - 2\mathbb{E}_x[e_{f,x}e_{h,x}]| \tag{D.48}$$

$$\leqslant e_{\hat{h}} \tag{D.49}$$

$$\leqslant \sup_{h\in\mathcal{H}} e_h \tag{D.50}$$

recovering the bound in the proxy risk paper.

The above calculation also shows that our bound is tighter. Their bound is only for the case when only one check model $\hat{h}$ is learned and also for the worst case. First, it is challenging to find an $\hat{h}$ with a small error in many practical scenarios. For

example, the test data contains outlier inputs which are not similar to the training data. It is then unlikely to find an $\hat{h}$ with small errors on these data points since no enough label information is available. However, it is still possible to have a good estimation of the accuracy, since the outlier data are different from the training data and thus can be detected, and we know that f is likely to make errors on them. Second, the bound is also too pessimistic. In the experiments, we observed that the proxy risk method can still achieve reasonable estimation (about 10% away from the true accuracy), even when the error of $\hat{h}$ is very large (e.g., $> 60\%$ while $\sup_{h \in \mathcal{H}} e_h$ is even larger).

Our lemma suggests that by allowing an ensemble $h \sim \mathcal{T}$ with proper diversity, we have more flexibility and can significantly improve the pessimistic bound. For the example given above, the ensemble method can potentially handle the outlier input data: for hypotheses agreeing with the training data, they are still likely to have disagreement on the outlier data and this disagreement thus reveals the potential error of f there, leading to an accurate estimation of the accuracy. We thus propose to use an ensemble method for estimating the accuracy.

Comparison with Calibration. A classic notion for uncertainty estimation is calibration of the machine learning model. It is well-known that if the pre-trained model f outputs confidence scores for class labels and the confidence is well-calibrated, then the average confidence approximates its accuracy. Unfortunately, it has been observed that many machine learning systems, in particular modern neural networks, are poorly calibrated, especially on test data with distribution shift (Guo et al., 2017; Ovadia et al., 2019) which is the most interesting case for accuracy estimation.

On the other hand, one can hope to obtain an ensemble of models that is well calibrated, such as the deep ensemble method (Lakshminarayanan et al., 2017). Below we show that a notion of well-calibration of the ensemble implies the agreement rate between the ensemble and the pre-trained model is a good estimation of the accuracy of the pre-trained model. Formally, we consider the simplified setting of perfect calibration defined as follows.

Definition D.5 (Perfect Calibration)**.** *An ensemble $\mathcal{T}$ of classifiers has perfect calibration on the dataset $U = \{(x, y_x)\}$ and function f, if for any class label $k \in \mathcal{Y}$ and any $p \in [0, 1]$,*

$$\Pr_{(x,y_x)\sim U}[y_x = k | C_k(x) = p, f(x) = k] = p. \tag{D.51}$$

where $C_k(x) := \Pr_{h\sim\mathcal{T}}[h(x) = k]$ is the confidence score of $\mathcal{T}$ for label k on the input x.

(The definition and the later analysis also applies to a classifier $C_k(x)$ outputting confidence scores, or replacing U with a data distribution.)

Proposition 14. *If the ensemble $\mathcal{T}$ has perfect calibration, then* $\mathrm{ar}(f, \mathcal{T}) = \mathrm{acc}(f)$.

Proof. By definition, we have

$$\begin{aligned}
\mathrm{ar}(f, \mathcal{T}) &= \Pr_{h,x}[h(x) = f(x)] && \text{(D.52)}\\
&= \mathbb{E}_x\left[\Pr_h[h(x) = f(x)]\right] && \text{(D.53)}\\
&= \mathbb{E}_x\left[C_{f(x)}(x)\right] && \text{(D.54)}\\
&= \mathbb{E}\left\{\mathbb{E}\left[C_{f(x)}(x) | C_k(x) = p, f(x) = k\right]\right\} && \text{(D.55)}\\
&= \mathbb{E}\{\mathbb{E}[p | C_k(x) = p, f(x) = k]\} && \text{(D.56)}\\
&= \mathbb{E}\{\mathbb{E}\{\Pr[y_x = k | C_k(x) = p, f(x) = k] | C_k(x) = p, f(x) = k\}\} && \text{(D.57)}\\
&= \mathbb{E}\{\Pr[y_x = k | C_k(x) = p, f(x) = k]\} && \text{(D.58)}\\
&= \Pr[y_x = f(x)] && \text{(D.59)}\\
&= \mathrm{acc}(f). && \text{(D.60)}
\end{aligned}$$

The third line follows from the definition of $C_k(x)$, the fourth line from the law of total expectation, the sixth line from perfect calibration, and the eighth line from the law of total expectation. □

On the other hand, our Lemma D.4 is more general: it shows even if the ensemble is not well calibrated, it is still possible for the agreement rate to be a good estimation of the accuracy. For illustration, consider a simple example with 4 points in U, and

x	x_2^-	x_1^-	x_1^+	x_2^+
y_x	$-$	$-$	$+$	$+$
$f(x)$	$-$	$+$	$-$	$+$
$h(x)$	$-$	$+$	$+$	$-$

Table D.1: An illustrative example showing even if the ensemble is not well calibrated, it is still possible for the agreement rate to be a good estimation of the accuracy.

only one model h from $\mathcal{T}$ (if $h(x) = k$, we view it as $\Pr_h[h(x) = k] = 1$). The predictions of f and h are shown in Table D.1. It is easy to see that h is not well-calibrated, e.g.,

$$\Pr_h[y_x = +|C_+(x) = 1] = \Pr_h[y_x = +|h(x) = +] = 1/2 \ll 1.$$

On the other hand, $\mathrm{ar}(f, \mathcal{T}) = \mathrm{acc}(f) = 1/2$. From the perspective of Lemma D.4, although the ensemble has a large error $e_{\mathcal{T}} = 1/2$, its predictions and those of f are properly correlated, such that

$$\mathrm{acc}(f) - \mathrm{ar}(f, \mathcal{T}) = e_{\mathcal{T}} - 2\mathbb{E}[e_{h,x} e_{f,x}] = \frac{1}{2} - 2 \cdot \frac{1}{4} = 0$$

leading to an accurate estimation of the accuracy of f.

E APPENDIX FOR CHAPTER 7

E.1 Baselines

We consider two selective classification baselines Softmax Response (SR) (Geifman and El-Yaniv, 2017) and Deep Ensembles (DE) (Lakshminarayanan et al., 2017) and combine them with active learning techniques. We describe them in detail below.

E.1.1 Softmax Response

Suppose the neural network classifier is f where the last layer is a softmax. Let $f(\mathbf{x} \mid k)$ be the soft response output for the k-th class. Then the classifier is defined as $f(\mathbf{x}) = \arg\max_{k \in \mathcal{Y}} f(\mathbf{x} \mid k)$ and the selection scoring function is defined as $g(\mathbf{x}) = \max_{k \in \mathcal{Y}} f(\mathbf{x} \mid k)$, which is also known as the Maximum Softmax Probability (MSP) of the neural network. Recall that with f and g, the selective classifier is defined in Eq (7.1). We use active learning to fine-tune the model f to improve selective prediction performance of SR on the unlabeled test dataset U_X. The complete algorithm is presented in Algorithm 5. In our experiments, we always set $\lambda = 1$. We use the joint training objective (E.9) to avoid over-fitting to the small labeled test set $\cup_{l=1}^{t} \tilde{B}_l$ and prevent the model from forgetting the source training knowledge. The algorithm can be combined with different kinds of acquisition functions. We describe the acquisition functions considered for SR below.

Uniform. In the t-th round of active learning, we select $[\frac{M}{T}]$ data points as the batch B_t from $U_X \setminus \cup_{l=0}^{t-1} B_l$ via uniform random sampling. The corresponding acquisition function is: $a(B, f_{t-1}, g_{t-1}) = 1$. When solving the objective (E.8), the tie is broken randomly.

Confidence. We define the confidence score of f on the input $\mathbf{x}$ as

$$S_{\text{conf}}(\mathbf{x}; f) = \max_{k \in \mathcal{Y}} \quad f(\mathbf{x} \mid k) \tag{E.1}$$

Then the acquisition function in the t-th round of active learning is defined as:

$$a(B, f_{t-1}, g_{t-1}) = -\sum_{\mathbf{x} \in B} S_{\text{conf}}(\mathbf{x}; f_{t-1}) \tag{E.2}$$

That is we select those test examples with the lowest confidence scores for labeling.

Entropy. We define the entropy score of f on the input $\mathbf{x}$ as

$$S_{\text{entropy}}(\mathbf{x}; f) = \sum_{k \in \mathcal{Y}} \quad -f(\mathbf{x} \mid k) \cdot \log f(\mathbf{x} \mid k) \tag{E.3}$$

Then the acquisition function in the t-th round of active learning is defined as:

$$a(B, f_{t-1}, g_{t-1}) = \sum_{\mathbf{x} \in B} S_{\text{entropy}}(\mathbf{x}; f_{t-1}) \tag{E.4}$$

That is we select those test examples with the highest entropy scores for labeling.

Margin. We define the margin score of f on the input $\mathbf{x}$ as

$$S_{\text{margin}}(\mathbf{x}; f) = f(\mathbf{x} \mid \hat{y}) - \max_{k \in \mathcal{Y} \setminus \{\hat{y}\}} \quad f(\mathbf{x} \mid k) \tag{E.5}$$

$$\text{s.t.} \quad \hat{y} = \arg\max_{k \in \mathcal{Y}} \quad f(\mathbf{x} \mid k) \tag{E.6}$$

Then the acquisition function in the t-th round of active learning is defined as:

$$a(B, f_{t-1}, g_{t-1}) = -\sum_{\mathbf{x} \in B} S_{\text{margin}}(\mathbf{x}; f_{t-1}) \tag{E.7}$$

That is we select those test examples with lowest margin scores for labeling.

kCG. We use the k-Center-Greedy algorithm proposed in Sener and Savarese (2017) to select test examples for labeling in each round.

CLUE. We use the Clustering Uncertainty-weighted Embeddings (CLUE) proposed in Prabhu et al. (2021) to select test examples for labeling in each round. Following Prabhu et al. (2021), we set the hyper-parameter $T = 0.1$ on DomainNet and set $T = 1.0$ on other datasets.

BADGE. We use the Diverse Gradient Embeddings (BADGE) proposed in Ash et al. (2019) to select test examples for labeling in each round.

Algorithm 5 Softmax Response with Active Learning

Require: A training dataset $\mathcal{D}^{\text{tr}}$, an unlabeled test dataset U_X, the number of rounds T, the labeling budget M, a source-trained model $\bar{f}$, an acquisition function a and a hyper-parameter λ.
Let $f_0 = \bar{f}$.
Let $B_0 = \emptyset$.
Let $g_t(\mathbf{x}) = \max_{k \in \mathcal{Y}} f_t(\mathbf{x} \mid k)$.
for $t = 1, \cdots, T$ **do**
Select a batch B_t with a size of $m = [\frac{M}{T}]$ from U_X for labeling via:

$$B_t = \arg \max_{B \subset U_X \setminus (\cup_{l=0}^{t-1} B_l), |B|=m} a(B, f_{t-1}, g_{t-1}) \tag{E.8}$$

Use an oracle to assign ground-truth labels to the examples in B_t to get $\tilde{B}_t$.
Fine-tune the model f_{t-1} using the following training objective:

$$\min_{\theta} \quad \mathbb{E}_{(\mathbf{x},y) \in \cup_{l=1}^{t} \tilde{B}_l} \quad \ell_{CE}(\mathbf{x}, y; \theta) + \lambda \cdot \mathbb{E}_{(\mathbf{x},y) \in \mathcal{D}^{\text{tr}}} \quad \ell_{CE}(\mathbf{x}, y; \theta) \tag{E.9}$$

where θ is the model parameters of f_{t-1} and ℓ_{CE} is the cross-entropy loss function.
Let $f_t = f_{t-1}$.
end for
Ensure: The classifier $f = f_T$ and the selection scoring function $g = \max_{k \in \mathcal{Y}} f(\mathbf{x} \mid k)$.

E.1.2 Deep Ensembles

It has been shown that deep ensembles can significantly improve selective prediction performance (Lakshminarayanan et al., 2017), not only because deep ensembles are more accurate than a single model, but also because deep ensembles yield more calibrated confidence.

Suppose the ensemble model f contains N models $f^1, \ldots, f^N$. Let $f^j(\mathbf{x} \mid k)$ denote the predicted probability of the model f^j on the k-th class. We define the predicted probability of the ensemble model f on the k-th class as:

$$f(\mathbf{x} \mid k) = \frac{1}{N} \sum_{j=1}^{N} f^j(\mathbf{x} \mid k). \tag{E.10}$$

The classifier is defined as $f(\mathbf{x}) = \arg\max_{k \in \mathcal{Y}} f(\mathbf{x} \mid k)$ and the selection scoring function is defined as $g(\mathbf{x}) = \max_{k \in \mathcal{Y}} f(\mathbf{x} \mid k)$. We use active learning to fine-tune each model f^j in the ensemble to improve selective prediction performance of the ensemble on the unlabeled test dataset U_X. Each model f^j is first initialized by the source-trained model $\bar{f}$, and then fine-tuned independently via Stochastic Gradient Decent (SGD) with different sources of randomness (e.g., different random order of the training batches) on the training dataset $\mathcal{D}^{\text{tr}}$ and the selected labeled test data. Note that this way to construct the ensembles is different from the standard Deep Ensembles method, which trains the models from different random initialization. We use this way to construct the ensemble due to the constraint in our problem setting, which requires us to fine-tune a given source-trained model $\bar{f}$. Training the models from different random initialization might lead to an ensemble with better performance, but it is much more expensive, especially when the training dataset and the model are large (e.g., training foundation models). Thus, the constraint in our problem setting is feasible in practice. The complete algorithm is presented in Algorithm 6. In our experiments, we always set $\lambda = 1$, $N = 5$, and $n_s = 1000$. We also use joint training here and the reasons are the same as those for the Softmax Response baseline. The algorithm can be combined with different kinds of acquisition functions. We describe the acquisition functions considered

below.

Uniform. In the t-th round of active learning, we select $[\frac{M}{T}]$ data points as the batch B_t from $U_X \setminus \cup_{l=0}^{t-1} B_l$ via uniform random sampling. The corresponding acquisition function is: $a(B, f_{t-1}, g_{t-1}) = 1$. When solving the objective (E.17), the tie is broken randomly.

Confidence. The confidence scoring function S_{conf} for the ensemble model f is the same as that in Eq. (E.1) ($f(\mathbf{x} \mid k)$ for the ensemble model f is defined in Eq. (E.10)). The acquisition function in the t-th round of active learning is defined as:

$$a(B, f_{t-1}, g_{t-1}) = -\sum_{\mathbf{x} \in B} S_{\text{conf}}(\mathbf{x}; f_{t-1}) \tag{E.11}$$

That is we select those test examples with the lowest confidence scores for labeling.

Entropy. The entropy scoring function S_{entropy} for the ensemble model f is the same as that in Eq. (E.3). The acquisition function in the t-th round of active learning is defined as:

$$a(B, f_{t-1}, g_{t-1}) = \sum_{\mathbf{x} \in B} S_{\text{entropy}}(\mathbf{x}; f_{t-1}), \tag{E.12}$$

That is we select those test examples with the highest entropy scores for labeling.

Margin. The margin scoring function S_{margin} for the ensemble model f is the same as that in Eq. (E.5). The acquisition function in the t-th round of active learning is defined as:

$$a(B, f_{t-1}, g_{t-1}) = -\sum_{\mathbf{x} \in B} S_{\text{margin}}(\mathbf{x}; f_{t-1}) \tag{E.13}$$

That is we select those test examples with the lowest margin scores for labeling.

Avg-KLD. The Average Kullback-Leibler Divergence (Avg-KLD) is proposed in McCallum et al. (1998) as a disagreement measure for the model ensembles, which can be used for sample selection in active learning. The Avg-KLD score of the ensemble model f on the input $\mathbf{x}$ is defined as:

$$S_{kl}(\mathbf{x}; f) = \frac{1}{N} \sum_{j=1}^{N} \sum_{k \in \mathcal{Y}} f^j(\mathbf{x} \mid k) \cdot \log \frac{f^j(\mathbf{x} \mid k)}{f(\mathbf{x} \mid k)}. \tag{E.14}$$

Then the acquisition function in the t-th round of active learning is defined as:

$$a(B, f_{t-1}, g_{t-1}) = \sum_{\mathbf{x} \in B} S_{kl}(\mathbf{x}; f_{t-1}), \tag{E.15}$$

That is we select those test examples with the highest Avg-KLD scores for labeling.

CLUE. CLUE (Prabhu et al., 2021) is proposed for a single model. Here, we adapt CLUE for the ensemble model, which requires a redefinition of the entropy function $\mathcal{H}(Y \mid \mathbf{x})$ and the embedding function $\phi(\mathbf{x})$ used in the CLUE algorithm. We define the entropy function as Eq. (E.3) with the ensemble model f. Suppose ϕ^j is the embedding function for the model f^j in the ensemble. Then, the embedding of the ensemble model f on the input $\mathbf{x}$ is $[\phi^1(\mathbf{x}), \ldots, \phi^N(\mathbf{x})]$, which is the concatenation of the embeddings of the models $f^1, \ldots, f^N$ on $\mathbf{x}$. Following Prabhu et al. (2021), we set the hyper-parameter $T = 0.1$ on DomainNet and set $T = 1.0$ on other datasets.

BADGE. BADGE (Ash et al., 2019) is proposed for a single model. Here, we adapt BADGE for the ensemble model, which requires a redefinition of the gradient embedding g_x in the BADGE algorithm. Towards this end, we propose the gradient embedding g_x of the ensemble model f as the concatenation of the gradient embeddings of the models $f^1, \ldots, f^N$.

Algorithm 6 Deep Ensembles with Active Learning

Require: A training dataset $\mathcal{D}^{\text{tr}}$, An unlabeled test dataset U_X, the number of rounds T, the total labeling budget M, a source-trained model $\bar{f}$, an acquisition function $a(B, f, g)$, the number of models in the ensemble N, the number of initial training steps n_s, and a hyper-parameter λ.
Let $f_0^j = \bar{f}$ for $j = 1, \ldots, N$.
Fine-tune each model f_0^j in the ensemble via SGD for n_s training steps independently using the following training objective with different randomness:

$$\min_{\theta^j} \quad \mathbb{E}_{(x,y)\in\mathcal{D}^{\text{tr}}} \quad \ell_{CE}(\mathbf{x}, y; \theta^j) \tag{E.16}$$

where θ^j is the model parameters of f_0^j and ℓ_{CE} is the cross-entropy loss function.

Let $B_0 = \emptyset$.
Let $g_t(\mathbf{x}) = \max_{k\in\mathcal{Y}} f_t(\mathbf{x} \mid k)$.
for $t = 1, \cdots, T$ **do**
Select a batch B_t with a size of $m = [\frac{M}{T}]$ from U_X for labeling via:

$$B_t = \arg \max_{B\subset U_X\setminus(\cup_{l=0}^{t-1}B_l), |B|=m} a(B, f_{t-1}, g_{t-1}) \tag{E.17}$$

Use an oracle to assign ground-truth labels to the examples in B_t to get $\tilde{B}_t$.
Fine-tune each model f_{t-1}^j in the ensemble via SGD independently using the following training objective with different randomness:

$$\min_{\theta^j} \quad \mathbb{E}_{(x,y)\in\cup_{l=1}^{t}\tilde{B}_l} \quad \ell_{CE}(\mathbf{x}, y; \theta^j) + \lambda \cdot \mathbb{E}_{(x,y)\in\mathcal{D}^{\text{tr}}} \quad \ell_{CE}(\mathbf{x}, y; \theta^j) \tag{E.18}$$

where θ^j is the model parameters of f_{t-1}^j.
Let $f_t^j = f_{t-1}^j$ for $j = 1, \ldots, N$.
end for
Ensure: The classifier $f = f_T$ and the selection scoring function $g = \max_{k\in\mathcal{Y}} f(\mathbf{x} \mid k)$.

E.2 Details of Experimental Setup

E.2.1 Datasets

We describe the datasets used below. For all image datasets, we normalize the range of pixel values to $[0,1]$.

MNIST→SVHN. The source training dataset $\mathcal{D}^{\text{tr}}$ is MNIST (LeCun, 1998) while the target test dataset $\mathcal{U}_X$ is SVHN (Netzer et al., 2011). MNIST consists 28×28 grayscale images of handwritten digits, containing in total 5,500 training images and 1,000 test images. We resize each image to be 32×32 resolution and change them to be colored. We use the training set of MNIST as $\mathcal{D}^{\text{tr}}$ and the test set of MNIST as the source validation dataset. SVHN consists 32×32 colored images of digits obtained from house numbers in Google Street View images. The training set has 73,257 images and the test set has 26,032 images. We use the test set of SVHN as $\mathcal{U}_X$.

CIFAR-10→CINIC-10. The source training dataset $\mathcal{D}^{\text{tr}}$ is CIFAR-10 (Krizhevsky et al., 2009) while the target test dataset $\mathcal{U}_X$ is CINIC-10 (Darlow et al., 2018). CIFAR-10 consists 32×32 colored images with ten classes (dogs, frogs, ships, trucks, etc.), each consisting of 5,000 training images and 1,000 test images. We use the training set of CIFAR-10 as $\mathcal{D}^{\text{tr}}$ and the test set of CIFAR-10 as the source validation dataset. During training, we apply random horizontal flipping and random cropping with padding data augmentations to the training images. CINIC-10 is an extension of CIFAR-10 via the addition of downsampled ImageNet images. CINIC-10 has a total of 270,000 images equally split into training, validation, and test. In each subset (90,000 images) there are ten classes (identical to CIFAR-10 classes). There are 9,000 images per class per subset. We use a subset of the CINIC-10 test set containing 30,000 images as $\mathcal{U}_X$.

FMoW. We use the FMoW-WILDS dataset from Koh et al. (2021). FMoW-wilds is based on the Functional Map of the World dataset (Christie et al., 2018), which

collected and categorized high-resolution satellite images from over 200 countries based on the functional purpose of the buildings or land in the image, over the years 2002–2018. The task is multi-class classification, where the input $\mathbf{x}$ is an RGB satellite image, the label y is one of 62 building or land use categories, and the domain d represents both the year the image was taken as well as its geographical region (Africa, the Americas, Oceania, Asia, or Europe). The training set contains 76,863 images from the years 2002-2013. The In-Distribution (ID) validation set contains 11,483 images from the years 2002-2013. The OOD test set contains 22,108 images from the years 2016-2018. We resize each image to be 96×96 resolution to save computational cost. We use the training set as $\mathcal{D}^{\text{tr}}$ and the ID validation set as the source validation dataset. During training, we apply random horizontal flipping and random cropping with padding data augmentations to the training images. We use the OOD test set as U_X.

Amazon Review. We use the Amazon Review WILDS dataset from Koh et al. (2021). The dataset comprises 539,502 customer reviews on Amazon taken from the Amazon Reviews dataset (Ni et al., 2019). The task is multi-class sentiment classification, where the input $\mathbf{x}$ is the text of a review, the label y is a corresponding star rating from 1 to 5, and the domain d is the identifier of the reviewer who wrote the review. The training set contains 245,502 reviews from 1,252 reviewers. The In-Distribution (ID) validation set contains 46,950 reviews from 626 of the 1,252 reviewers in the training set. The Out-Of-Distribution (OOD) test set contains 100,050 reviews from another set of 1,334 reviewers, distinct from those of the training set. We use the training set as $\mathcal{D}^{\text{tr}}$ and the ID validation set as the source validation dataset. We use a subset of the OOD test set containing 22,500 reviews from 300 reviewers as U_X.

DomainNet. DomainNet (Peng et al., 2019) is a dataset of common objects in six different domains. All domains include 345 categories (classes) of objects such as Bracelet, plane, bird, and cello. We use five domains from DomainNet including: (1) Real: photos and real world images. The training set from the Real domain

has 120,906 images while the test set has 52,041 images; (2) Clipart: a collection of clipart images. The training set from the Clipart domain has 33,525 images while the test set has 14,604 images; (3) Sketch: sketches of specific objects. The training set from the Sketch has 48,212 images while the test set has 20,916 images; (4) Painting: artistic depictions of objects in the form of paintings. The training set from the Painting domain has 50,416 images while the test set has 21,850 images. (5) Infograph: infographic images with specific objects. The training set from the Infograph domain has 36,023 images while the test set has 15,582 images. We resize each image from all domains to be 96×96 resolution to save computational cost. We use the training set from the Real domain as $\mathcal{D}^{\text{tr}}$ and the test set from the Real domain as the source validation dataset. During training, we apply random horizontal flipping and random cropping with padding data augmentations to the training images. We use the test sets from three domains Clipart, Sketch, and Painting as three different $\mathcal{U}_X$ for evaluation. So we evaluate three shifts: Real→Clipart (R→C), Real→Sketch (R→S), and Real→Painting (R→P). We use the remaining shift Real→Infograph (R→I) as a validation dataset for tuning the hyper-parameters.

Otto. The Otto Group Product Classification Challenge (Benjamin Bossan, 2015) is a tabular dataset hosted on Kaggle [1]. The task is to classify each product with 93 features into 9 categories. Each target category represents one of the most important product categories (like fashion, electronics, etc). It contains 61,878 training data points. Since it only provides labels for the training data, we need to create the training, validation and test set. To create a test set that is from a different distribution than the training set, we apply the Local Outlier Factor (LOF) (Breunig et al., 2000), which is an unsupervised outlier detection method, on the Otto training data to identify a certain fraction (e.g., 0.2) of outliers as the test set. Specifically, we apply the *LocalOutlierFactor* function provided by scikit-learn (Pedregosa et al., 2011) on the training data with a contamination of 0.2 (contamination value determines the proportion of outliers in the data set) to

[1] `https://kaggle.com/competitions/otto-group-product-classification-challenge`

identify the outliers. We identify 12,376 outlier data points and use them as the test set $\mathcal{U}_X$. We then randomly split the remaining data into a training set $\mathcal{D}^{\text{tr}}$ with 43,314 data points and a source validation set with 6,188 data points. We show that the test set indeed has a distribution shift compared to the source validation set, which causes the model trained on the training set to have a drop in performance (see Table E.1 in Appendix E.3.1).

E.2.2 Details on Model Architectures and Training on Source Data

On all datasets, we use the following supervised training objective for training models on the source training set $\mathcal{D}^{\text{tr}}$:

$$\min_{\theta} \quad \mathbb{E}_{(x,y)\in\mathcal{D}^{\text{tr}}} \quad \ell_{CE}(x, y; \theta) \tag{E.19}$$

where ℓ_{CE} is the cross-entropy loss and θ is the model parameters.

On MNIST→SVHN, we use the Convolutional Neural Network (CNN) (LeCun et al., 1989) consisting of four convolutional layers followed by two fully connected layers with batch normalization and dropout layers. We train the model on the training set of MNIST for 20 epochs using the Adam optimizer (Kingma and Ba, 2014) with a learning rate of 10^{-3} and a batch size of 128.

On CIFAR-10→CINIC-10, we use the ResNet-20 network (He et al., 2016b). We train the model on the training set of CIFAR-10 for 200 epochs using the SGD optimizer with a learning rate of 0.1, a momentum of 0.9, and a batch size of 128. The learning rate is multiplied by 0.1 at the 80, 120, and 160 epochs, respectively, and is multiplied by 0.5 at the 180 epoch.

On the FMoW dataset, we use the DensetNet-121 network (Huang et al., 2017b) pre-trained on ImageNet. We train the model further for 50 epochs using the Adam optimizer with a learning rate of 10^{-4} and a batch size of 128.

On the Amazon Review dataset, we use the pre-trained RoBERTa model (Liu et al., 2019) as the base model to extract the embedding of the input sentence for

classification (i.e., RoBERTa's output for the [CLS] token) and then build an eight-layer fully connected neural network (also known as a multi-layer perceptron) with batch normalization, dropout layers and L2 regularization on top of the embedding. Note that we only update the parameters of the fully connected neural network without updating the parameters of the pre-trained RoBERTa base model during training (i.e., freeze the parameters of the RoBERTa base model during training). We train the model for 200 epochs using the Adam optimizer with a learning rate of 10^{-3} and a batch size of 128.

On the DomainNet dataset, we use the ResNet-50 network (He et al., 2016a) pre-trained on ImageNet. We train the model further on the training set from the Real domain for 50 epochs using the Adam optimizer with a learning rate of 10^{-4} and a batch size of 128.

On the Otto dataset, we use a six-layer fully connected neural network (also known as a multi-layer perceptron) with batch normalization, dropout layers and L2 regularization. We train the model on the created training set for 200 epochs using the Adam optimizer with a learning rate of 10^{-3} and a batch size of 128.

E.2.3 Active learning hyper-parameters

During the active learning process, we fine-tune the model on the selected labeled test data. During fine-tuning, we don't apply any data augmentation to the test data. We use the same fine-tuning hyper-parameters for different methods to ensure a fair comparison. The optimizer used is the same as that in the source training stage (described in Appendix E.2.2). On MNIST→SVHN, we use a learning rate of 10^{-3}; On CIFAR-10→CINIC-10, we use a learning rate of 5×10^{-3}; On FMoW, we use a learning rate of 10^{-4}; On Amazon Review, we use a learning rate of 10^{-3}; On DomainNet, we use a learning rate of 10^{-4}; On Otto, we use a learning rate of 10^{-3}. On all datasets, we fine-tune the model for at least 50 epochs and up to 200 epochs with a batch size of 128 and early stopping using 10 patient epochs.

E.3 Additional Experimental Results

E.3.1 Evaluate Source-Trained Models

In this section, we evaluate the accuracy of the source-trained models on the source validation dataset and the target test dataset U_X. The models are trained on the source training set $\mathcal{D}^{\text{tr}}$ (refer to Appendix E.2.2 for the details of source training). The source validation data are randomly sampled from the training data distribution while the target test data are sampled from a different distribution than the training data distribution. The results in Table E.1 show that the models trained on $\mathcal{D}^{\text{tr}}$ always suffer a drop in accuracy when evaluating them on the target test dataset U_X.

Dataset	Source Accuracy	Target Accuracy
MNIST→SVHN	99.40	24.68
CIFAR-10→CINIC-10	90.46	71.05
FMoW	46.25	38.01
Amazon Review	65.39	61.40
DomainNet (R→C)	63.45	33.37
DomainNet (R→P)	63.45	26.29
DomainNet (R→S)	63.45	16.00
Otto	80.72	66.09

Table E.1: Results of evaluating the accuracy of the source-trained models on the source validation dataset and the target test dataset U_X. All numbers are percentages.

E.3.2 Complete Evaluation Results

We give complete experimental results for the baselines and the proposed method ASPEST on all datasets in this section. We repeat each experiment three times with different random seeds and report the mean and standard deviation (std) values. These results are shown in Table E.2 (MNIST→SVHN), Table E.3 (CIFAR-10→CINIC-10), Table E.4 (FMoW), Table E.5 (Amazon Review), Table E.6 (DomainNet R→C), Table E.7 (DomainNet R→P), Table E.8 (DomainNet R→S) and

Table E.9 (Otto). Our results show that the proposed method ASPEST consistently outperforms the baselines across different image, text and structured datasets.

Dataset	MNIST→SVHN								
Metric	cov|acc $\geqslant$ 90% ↑			acc|cov $\geqslant$ 90% ↑			AUC ↑		
Labeling Budget	100	500	1000	100	500	1000	100	500	1000
SR+Uniform	0.00±0.0	51.46±3.7	75.57±0.9	58.03±1.5	76.69±1.2	84.39±0.2	74.08±1.5	88.80±0.8	93.57±0.2
SR+Confidence	0.00±0.0	55.32±5.1	82.22±1.3	47.66±3.4	79.02±0.7	87.19±0.4	64.14±2.8	89.93±0.6	94.62±0.2
SR+Entropy	0.00±0.0	0.00±0.0	75.08±2.4	47.93±7.0	77.09±1.0	84.81±0.7	65.88±4.7	88.19±0.8	93.37±0.5
SR+Margin	0.00±0.0	63.60±2.7	82.19±0.3	61.39±0.5	80.96±0.9	86.97±0.2	76.79±0.5	91.24±0.5	94.82±0.1
SR+kCG	2.52±1.3	23.04±0.3	38.97±2.6	34.57±4.4	52.76±1.1	64.34±4.8	48.83±7.2	73.65±1.0	83.16±2.0
SR+CLUE	0.00±0.0	62.03±2.4	81.29±1.1	57.35±1.9	79.55±0.8	86.28±0.5	72.72±1.9	90.98±0.5	94.99±0.2
SR+BADGE	0.00±0.0	62.55±4.4	82.39±2.8	59.82±1.7	79.49±1.6	86.96±0.9	76.06±1.6	91.09±0.9	95.16±0.6
DE+Uniform	24.71±5.6	68.98±1.6	83.67±0.1	63.22±1.7	81.67±0.4	87.32±0.1	79.36±1.7	92.47±0.2	95.48±0.0
DE+Entropy	6.24±8.8	63.30±6.5	84.62±1.5	56.61±0.6	80.16±2.0	88.05±0.5	72.51±1.5	91.21±1.4	95.45±0.5
DE+Confidence	14.92±5.1	67.87±1.4	89.41±0.3	61.11±2.9	81.80±0.5	89.75±0.1	75.85±3.0	92.16±0.2	96.19±0.1
DE+Margin	21.59±3.8	77.84±2.8	92.75±0.3	62.88±1.2	85.11±1.1	91.17±0.1	78.59±1.4	94.31±0.6	97.00±0.0
DE+Avg-KLD	10.98±4.6	61.45±3.4	88.06±2.2	54.80±1.6	78.21±1.6	89.23±0.9	71.67±2.2	90.92±0.8	96.23±0.4
DE+CLUE	22.34±1.4	69.23±1.9	82.80±1.0	59.47±1.3	81.05±0.9	86.78±0.4	76.88±1.0	92.70±0.5	95.56±0.2
DE+BADGE	22.02±4.5	72.31±1.2	88.23±0.4	61.23±1.9	82.69±0.5	89.15±0.2	77.65±1.9	93.38±0.2	96.51±0.1
ASPEST (ours)	**52.10**±4.0	**89.22**±0.9	**98.70**±0.4	**76.10**±1.5	**89.62**±0.4	**93.92**±0.3	**88.84**±1.0	**96.62**±0.2	**98.06**±0.1

Table E.2: Results of comparing ASPEST to the baselines on MNIST→SVHN. The mean and std of each metric over three random runs are reported (mean±std). All numbers are percentages. **Bold** numbers are superior results.

E.3.3 Effect of combining selective prediction with active learning

Selective prediction without active learning corresponds to the case where the labeling budget $M = 0$ and the selected set $B^* = \emptyset$. To make fair comparisons with selective prediction methods without active learning, we define a new coverage metric:

$$\mathrm{cov}^*(f_s, \tau) = \mathbb{E}_{\mathbf{x} \sim U_X} \mathbb{I}[g(\mathbf{x}) \geqslant \tau \wedge \mathbf{x} \notin B^*] \tag{E.20}$$

The range of $\mathrm{cov}^*(f_s, \tau)$ is $[0, 1 - \frac{M}{n}]$, where $M = |B^*|$ and $n = |U_X|$. If we use a larger labeling budget M for active learning, then the upper bound of $\mathrm{cov}^*(f_s, \tau)$ will be smaller. Thus, in order to beat selective classification methods without active learning, active selective prediction methods need to use a small labeling

Dataset	CIFAR-10→CINIC-10								
Metric	$\mathrm{cov}\vert\mathrm{acc} \geqslant 90\% \uparrow$			$\mathrm{acc}\vert\mathrm{cov} \geqslant 90\% \uparrow$			AUC ↑		
Labeling Budget	500	1000	2000	500	1000	2000	500	1000	2000
SR+Uniform	57.43±0.2	57.15±0.6	58.37±0.7	75.67±0.2	75.69±0.1	76.11±0.3	89.77±0.0	89.81±0.1	90.09±0.2
SR+Confidence	57.96±0.6	57.05±0.7	61.11±1.1	76.49±0.2	76.87±0.2	78.77±0.4	90.00±0.2	89.92±0.2	90.91±0.3
SR+Entropy	57.78±0.7	57.07±1.4	61.07±0.4	76.57±0.3	76.71±0.5	78.85±0.2	90.01±0.2	89.94±0.3	90.88±0.0
SR+Margin	57.72±0.8	57.98±0.7	61.71±0.2	76.24±0.2	76.90±0.2	78.42±0.2	89.95±0.2	90.14±0.1	91.02±0.0
SR+kCG	57.90±0.5	57.81±0.7	60.36±0.3	75.59±0.1	75.73±0.2	76.68±0.2	89.78±0.1	89.79±0.2	90.41±0.2
SR+CLUE	57.29±0.5	58.89±0.5	62.28±0.2	75.74±0.2	76.68±0.3	78.10±0.2	89.67±0.2	90.15±0.1	91.03±0.1
SR+BADGE	58.58±0.6	58.63±0.3	61.95±0.4	76.33±0.5	76.58±0.1	78.26±0.2	90.05±0.2	90.16±0.1	90.99±0.0
DE+Uniform	58.06±0.3	58.72±0.1	59.54±0.3	76.65±0.1	77.06±0.2	77.46±0.1	90.26±0.1	90.45±0.1	90.73±0.1
DE+Entropy	58.91±0.6	60.96±0.2	63.85±0.2	77.66±0.1	79.14±0.1	80.82±0.2	90.55±0.1	91.16±0.1	91.89±0.0
DE+Confidence	58.53±0.3	61.03±0.6	64.42±0.2	77.73±0.2	79.00±0.1	80.87±0.0	90.53±0.0	91.11±0.1	91.96±0.0
DE+Margin	58.76±0.5	61.60±0.5	64.92±0.5	77.61±0.2	78.91±0.1	80.59±0.1	90.56±0.1	91.11±0.1	91.98±0.1
DE+Avg-KLD	59.99±0.6	62.05±0.3	65.02±0.5	77.84±0.1	79.15±0.0	81.04±0.1	90.74±0.1	91.30±0.1	92.10±0.1
DE+CLUE	59.27±0.1	61.16±0.4	64.42±0.0	77.19±0.1	78.37±0.2	79.44±0.1	90.44±0.1	91.03±0.1	91.74±0.0
DE+BADGE	59.37±0.4	61.61±0.1	64.53±0.4	77.13±0.1	78.33±0.2	79.44±0.3	90.49±0.1	91.12±0.0	91.78±0.1
ASPEST (ours)	**60.38**±0.3	**63.34**±0.2	**66.81**±0.3	**78.23**±0.1	**79.49**±0.1	**81.25**±0.1	**90.95**±0.0	**91.60**±0.0	**92.33**±0.1

Table E.3: Results of comparing ASPEST to the baselines on CIFAR-10→CINIC-10. The mean and std of each metric over three random runs are reported (mean±std). All numbers are percentages. **Bold** numbers are superior results.

budget to achieve significant accuracy and coverage improvement. We still use the accuracy metric defined in (7.4). We then define a new maximum accuracy at a target coverage t_c as:

$$\max_{\tau} \quad \mathrm{acc}(f_s, \tau), \quad \text{s.t.} \quad \mathrm{cov}^*(f_s, \tau) \geqslant t_c \tag{E.21}$$

We denote this metric as $\mathrm{acc}\vert\mathrm{cov}^* \geqslant t_c$.

We define a new maximum coverage at a target accuracy t_a metric as:

$$\max_{\tau} \quad \mathrm{cov}^*(f_s, \tau), \quad \text{s.t.} \quad \mathrm{acc}(f_s, \tau) \geqslant t_a \tag{E.22}$$

We denote this metric as $\mathrm{cov}^*\vert\mathrm{acc} \geqslant t_a$.

The results under these new metrics are shown in Table 7.2 (MNIST→SVHN), Table E.10 (CIFAR-10→CINIC-10 and Otto), Table E.11 (FMoW and Amazon Review) and Table E.12 (DomainNet). The results show that combining selective prediction with active learning can significantly improve the accuracy and coverage metrics, even with small labeling budgets.

Dataset	FMoW								
Metric	cov|acc $\geqslant$ 70% ↑			acc|cov $\geqslant$ 70% ↑			AUC ↑		
Labeling Budget	500	1000	2000	500	1000	2000	500	1000	2000
SR+Uniform	38.50±0.7	42.00±0.5	52.34±1.1	51.76±0.7	54.27±0.2	60.31±0.7	65.75±0.4	67.67±0.3	72.73±0.3
SR+Confidence	37.34±0.3	42.28±1.2	53.72±0.7	52.24±0.1	55.52±0.5	61.76±0.4	65.57±0.1	68.03±0.5	73.14±0.5
SR+Entropy	37.42±0.3	42.08±0.2	51.18±0.4	51.74±0.4	54.94±0.2	60.62±0.2	65.31±0.2	68.00±0.1	71.99±0.2
SR+Margin	38.40±1.4	44.67±0.7	55.68±1.5	52.88±0.3	56.66±0.4	62.98±0.7	66.11±0.6	69.12±0.4	73.86±0.5
SR+kCG	36.50±0.8	39.76±1.2	45.87±0.6	49.36±0.7	51.45±0.5	55.47±0.1	64.34±0.5	66.21±0.6	69.63±0.2
SR+CLUE	38.65±0.7	44.50±1.8	54.71±0.5	52.23±0.4	55.54±1.0	61.13±0.4	65.78±0.3	68.76±0.9	73.80±0.1
SR+BADGE	40.47±1.5	45.65±1.2	57.59±0.4	53.08±1.0	56.63±0.3	63.57±0.2	66.74±0.8	69.43±0.6	74.76±0.2
DE+Uniform	44.74±0.4	51.57±1.1	61.92±0.4	56.39±0.5	60.01±0.5	65.74±0.2	69.44±0.3	72.48±0.5	77.02±0.1
DE+Entropy	43.76±0.3	50.52±1.4	62.73±0.4	56.29±0.3	60.31±0.3	66.53±0.2	69.02±0.1	72.10±0.5	76.65±0.2
DE+Confidence	45.23±0.6	50.11±0.9	64.29±0.3	57.18±0.4	60.46±0.3	67.46±0.0	69.80±0.3	72.11±0.4	77.37±0.1
DE+Margin	46.35±0.6	54.79±1.3	69.70±0.8	57.84±0.3	62.43±0.5	69.87±0.4	70.18±0.3	73.62±0.3	78.88±0.4
DE+Avg-KLD	46.29±0.3	53.63±0.8	68.18±0.9	57.75±0.4	61.60±0.3	69.11±0.4	70.16±0.1	73.09±0.2	78.48±0.3
DE+CLUE	45.22±0.2	49.97±0.3	58.05±0.5	56.39±0.1	59.05±0.1	63.23±0.4	69.53±0.0	71.95±0.1	75.72±0.3
DE+BADGE	47.39±0.7	53.83±0.7	66.45±0.8	57.71±0.4	61.16±0.2	68.13±0.4	70.59±0.4	73.40±0.3	78.66±0.1
ASPEST (ours)	**53.05**±0.4	**59.86**±0.4	**76.52**±0.6	**61.18**±0.2	**65.18**±0.2	**72.86**±0.3	**71.12**±0.2	**74.25**±0.2	**79.93**±0.1

Table E.4: Results of comparing ASPEST to the baselines on FMoW. The mean and std of each metric over three random runs are reported (mean±std). All numbers are percentages. **Bold** numbers are superior results.

E.3.4 Empirical analysis for checkpoint ensemble

In this section, we analyze why the proposed checkpoint ensemble can improve selective prediction performance. We postulate the rationales: (1) the checkpoint ensemble can help with generalization; (2) the checkpoint ensemble can help with reducing overconfident wrong predictions.

Regarding (1), when fine-tuning the model on the small set of selected labeled test data, we hope that the fine-tuned model could generalize to remaining unlabeled test data. However, since the selected test set is small, we might have an overfitting issue. So possibly some intermediate checkpoints along the training path achieve better generalization than the end checkpoint. By using checkpoint ensemble, we might get an ensemble that achieves better generalization to remaining unlabeled test data. Although standard techniques like cross-validation and early stopping can also reduce overfitting, they are not suitable in the active selective prediction setup since the amount of labeled test data is small.

Regarding (2), when fine-tuning the model on the small set of selected labeled test data, the model can get increasingly confident on the test data. Since there exist

Dataset	Amazon Review								
Metric	cov|acc $\geqslant$ 80% $\uparrow$			acc|cov $\geqslant$ 80% $\uparrow$			AUC $\uparrow$		
Labeling Budget	500	1000	2000	500	1000	2000	500	1000	2000
SR+Uniform	13.71±11.3	24.10±5.3	24.87±2.6	65.13±0.8	66.33±0.6	66.26±0.3	72.71±1.5	73.64±0.7	73.53±0.7
SR+Confidence	11.28±8.9	17.96±4.0	33.19±1.4	65.15±0.7	66.29±0.4	68.94±0.1	72.89±0.7	73.25±0.7	76.17±0.2
SR+Entropy	5.55±7.8	13.32±9.5	25.47±1.8	65.11±1.1	66.56±0.7	67.31±0.7	71.96±1.6	72.53±1.1	74.19±0.5
SR+Margin	14.48±10.9	22.61±4.2	28.35±6.1	65.75±0.5	66.31±0.4	68.15±0.4	73.25±1.0	73.65±0.5	75.17±0.8
SR+kCG	20.02±11.0	17.02±12.2	29.08±4.2	64.03±3.1	66.17±0.5	66.63±1.0	72.34±3.2	74.35±0.7	74.49±1.0
SR+CLUE	4.15±5.9	25.15±4.9	31.88±2.1	66.17±0.4	66.30±0.4	67.12±0.7	73.43±0.4	74.07±0.7	75.29±0.9
SR+BADGE	22.58±0.4	23.78±6.4	30.71±4.6	66.29±0.4	66.31±0.6	68.58±0.7	73.80±0.6	74.00±1.0	75.76±0.8
DE+Uniform	34.35±1.4	33.15±1.1	36.55±1.8	68.13±0.4	68.12±0.6	68.88±0.2	76.20±0.3	76.16±0.4	77.07±0.3
DE+Entropy	31.74±1.4	36.29±1.6	40.33±1.7	68.19±0.3	69.44±0.2	71.27±0.3	75.98±0.4	77.10±0.3	78.53±0.3
DE+Confidence	35.12±1.8	34.48±1.4	40.46±0.5	69.07±0.3	69.47±0.2	71.08±0.2	76.63±0.2	76.87±0.3	78.27±0.1
DE+Margin	33.42±1.3	35.03±1.3	41.20±0.4	68.45±0.3	69.30±0.2	70.88±0.1	76.18±0.2	76.91±0.3	78.31±0.1
DE+Avg-KLD	33.03±1.5	38.55±3.2	41.75±1.8	68.63±0.3	69.95±0.4	71.10±0.3	76.21±0.4	77.62±0.6	78.62±0.3
DE+CLUE	33.92±3.0	35.27±1.4	34.83±3.1	68.09±0.3	68.07±0.3	68.40±0.6	76.27±0.6	76.65±0.3	76.69±0.7
DE+BADGE	32.23±3.7	36.18±1.5	40.58±3.3	68.34±0.4	68.87±0.2	70.29±0.3	76.13±0.7	77.09±0.2	78.44±0.5
ASPEST (ours)	**38.44**±0.7	**40.96**±0.8	**45.77**±0.1	**69.31**±0.3	**70.17**±0.2	**71.60**±0.2	**77.69**±0.1	**78.35**±0.2	**79.51**±0.2

Table E.5: Results of comparing ASPEST to the baselines on Amazon Review. The mean and std of each metric over three random runs are reported (mean±std). All numbers are percentages. **Bold** numbers are superior results.

high-confidence mis-classified test points, incorporating intermediate checkpoints along the training path into the ensemble can reduce the average confidence of the ensemble on those mis-classified test points. By using checkpoint ensemble, we might get an ensemble that has better confidence estimation for selective prediction on the test data.

We perform experiments on the image dataset MNIST→SVHN and the text dataset Amazon Review to verify these two hypotheses. We employ one-round active learning with a labeling budget of 100 samples. We use the margin sampling method for sample selection and fine-tune a single model on the selected labeled test data for 200 epochs. We first evaluate the median confidence of the model on the correctly classified and mis-classified test data respectively when fine-tuning the model on the selected labeled test data. In Figure E.1, we show that during fine-tuning, the model gets increasingly confident not only on the correctly classified test data, but also on the mis-classified test data.

We then evaluate the Accuracy, the area under the receiver operator characteristic curve (AUROC) and the area under the accuracy-coverage curve (AUC) metrics of the checkpoints during fine-tuning and the checkpoint ensemble constructed

Dataset	DomainNet R→C (easy)								
Metric	cov|acc $\geqslant$ 80% ↑			acc|cov $\geqslant$ 80% ↑			AUC ↑		
Labeling Budget	500	1000	2000	500	1000	2000	500	1000	2000
SR+Uniform	25.56±0.6	27.68±0.8	29.86±0.0	43.63±0.4	45.57±0.3	47.27±0.4	63.31±0.4	65.11±0.5	66.70±0.2
SR+Confidence	25.96±0.2	27.80±1.2	32.51±1.3	44.90±0.8	47.26±0.4	52.04±0.8	64.20±0.6	65.88±0.6	69.70±0.7
SR+Entropy	25.44±1.0	27.79±0.4	33.51±1.1	44.46±0.7	46.96±0.3	52.25±0.5	63.52±0.6	65.72±0.2	70.03±0.5
SR+Margin	26.28±1.2	27.77±1.0	32.92±0.4	45.24±1.0	47.12±0.7	52.29±0.4	64.37±0.8	65.91±0.6	70.01±0.4
SR+kCG	21.12±0.3	21.79±0.4	23.43±0.5	39.19±0.1	40.59±0.4	41.11±0.3	58.88±0.0	60.11±0.4	60.89±0.1
SR+CLUE	27.17±0.8	29.78±0.8	34.82±0.6	44.57±0.7	46.79±0.1	49.70±0.3	64.38±0.6	66.47±0.3	69.59±0.1
SR+BADGE	27.78±0.8	30.78±0.6	36.00±0.6	45.36±0.6	48.43±0.6	53.00±0.4	64.90±0.5	67.56±0.4	71.39±0.4
DE+Uniform	30.82±0.8	33.05±0.4	36.80±0.2	48.19±0.3	50.09±0.3	52.98±0.5	67.60±0.4	69.31±0.3	71.64±0.4
DE+Entropy	29.13±0.9	34.07±0.3	40.82±0.3	48.67±0.4	51.66±0.2	57.81±0.2	67.48±0.3	70.05±0.2	74.64±0.2
DE+Confidence	29.90±0.8	33.73±0.2	40.80±0.2	48.60±0.3	52.03±0.3	58.43±0.1	67.45±0.3	70.19±0.2	74.80±0.1
DE+Margin	31.82±1.3	35.68±0.2	43.39±0.7	50.12±0.4	53.19±0.4	59.17±0.2	68.85±0.4	71.29±0.3	75.79±0.3
DE+Avg-KLD	32.23±0.2	36.09±0.6	44.00±0.5	49.81±0.3	53.38±0.3	58.93±0.1	68.73±0.2	71.40±0.2	75.73±0.2
DE+CLUE	30.80±0.3	33.04±0.4	35.52±0.2	48.56±0.3	49.91±0.3	51.40±0.2	67.82±0.2	69.10±0.2	70.62±0.2
DE+BADGE	30.16±1.3	36.18±0.3	43.34±0.3	49.78±0.3	53.26±0.1	58.65±0.4	68.46±0.3	71.35±0.2	75.37±0.3
ASPEST (ours)	**37.38**±0.1	**39.98**±0.3	**48.29**±1.0	**54.56**±0.3	**56.95**±0.1	**62.69**±0.2	**71.61**±0.2	**73.27**±0.2	**77.40**±0.4

Table E.6: Results of comparing ASPEST to the baselines on DomainNet R→C. The mean and std of each metric over three random runs are reported (mean±std). All numbers are percentages. **Bold** numbers are superior results.

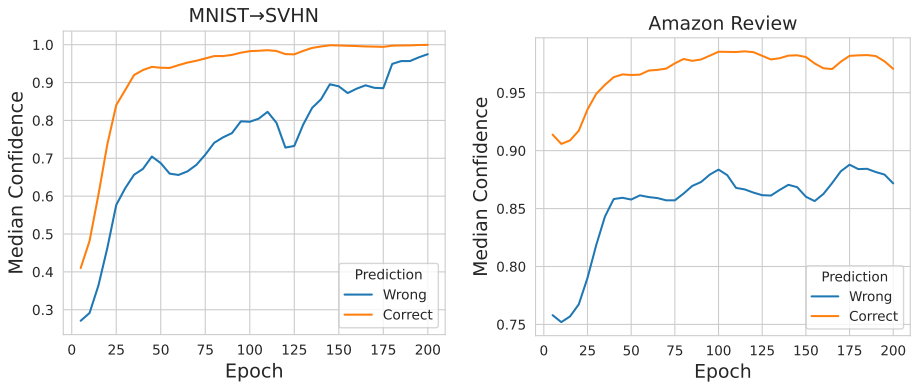


Figure E.1: Evaluating the median confidence of the model on the correctly classified and mis-classified test data respectively when fine-tuning the model on the selected labeled test data.

after fine-tuning on the target test dataset. The AUROC metric is equivalent to the probability that a randomly chosen correctly classified input has a higher confidence score than a randomly chosen mis-classified input. Thus, the AUROC metric can measure the quality of the confidence score for selective prediction. The results in Figure E.2 show that in the fine-tuning path, different checkpoints have different

Dataset	DomainNet R→P (medium)								
Metric	cov|acc $\geqslant$ 70% ↑			acc|cov $\geqslant$ 70% ↑			AUC ↑		
Labeling Budget	500	1000	2000	500	1000	2000	500	1000	2000
SR+Uniform	21.01±1.0	21.35±0.3	22.64±0.5	36.78±0.6	37.18±0.2	38.20±0.4	51.87±0.7	52.31±0.0	53.34±0.4
SR+Confidence	20.64±0.6	22.15±0.8	23.60±0.6	37.01±0.3	38.46±0.7	40.23±0.4	51.77±0.3	53.33±0.8	54.80±0.5
SR+Entropy	20.76±0.7	22.11±0.3	23.56±0.3	37.09±0.2	38.38±0.3	40.30±0.1	51.86±0.4	53.29±0.3	54.81±0.2
SR+Margin	21.43±0.4	23.29±0.3	24.70±1.0	37.21±0.2	39.15±0.4	40.81±0.4	52.33±0.1	54.09±0.3	55.70±0.4
SR+kCG	17.33±0.4	17.62±0.2	18.49±0.2	33.97±0.3	34.12±0.1	34.36±0.1	48.61±0.5	48.65±0.2	49.25±0.2
SR+CLUE	21.15±0.6	22.49±0.5	24.84±0.7	36.96±0.2	37.93±0.5	39.31±0.4	51.97±0.4	53.20±0.5	54.84±0.5
SR+BADGE	20.07±0.3	22.21±0.5	24.92±0.2	36.10±0.1	38.11±0.4	40.40±0.5	50.99±0.0	53.10±0.4	55.40±0.4
DE+Uniform	25.42±0.2	26.38±0.2	28.83±0.3	40.83±0.1	41.66±0.2	43.93±0.2	55.86±0.1	56.62±0.1	58.80±0.2
DE+Entropy	25.74±0.4	27.11±0.4	30.39±0.1	41.34±0.1	42.92±0.3	45.92±0.3	56.06±0.2	57.51±0.3	60.10±0.2
DE+Confidence	25.69±0.4	27.38±0.7	30.47±0.1	41.45±0.2	43.12±0.3	45.88±0.1	56.13±0.2	57.68±0.3	60.20±0.2
DE+Margin	25.78±0.3	27.88±0.5	31.03±0.4	41.26±0.2	43.13±0.3	46.23±0.4	56.23±0.2	57.90±0.3	60.49±0.3
DE+Avg-KLD	26.30±0.7	28.00±0.1	31.97±0.2	41.80±0.3	43.17±0.1	46.32±0.2	56.65±0.3	57.99±0.1	60.82±0.2
DE+CLUE	25.38±0.6	26.65±0.4	27.89±0.1	40.86±0.3	41.62±0.2	42.46±0.1	55.79±0.4	56.65±0.2	57.71±0.1
DE+BADGE	26.27±0.7	27.69±0.1	31.84±0.2	42.02±0.6	43.41±0.2	46.37±0.1	56.67±0.5	58.03±0.1	60.84±0.1
ASPEST (ours)	**29.69**±0.1	**32.50**±0.3	**35.46**±0.6	**44.96**±0.1	**46.77**±0.2	**49.42**±0.1	**58.74**±0.0	**60.36**±0.0	**62.84**±0.2

Table E.7: Results of comparing ASPEST to the baselines on DomainNet R→P. The mean and std of each metric over three random runs are reported (mean±std). All numbers are percentages. **Bold** numbers are superior results.

target test accuracy and the end checkpoint may not have the optimal target test accuracy. The checkpoint ensemble can have better target test accuracy than the end checkpoint. Also, in the fine-tuning path, different checkpoints have different confidence estimation (the quality of confidence estimation is measured by the metric AUROC) on the target test data and the end checkpoint may not have the optimal confidence estimation. The checkpoint ensemble can have better confidence estimation than the end checkpoint. Furthermore, in the fine-tuning path, different checkpoints have different selective prediction performance (measured by the metric AUC) on the target test data and the end checkpoint may not have the optimal selective prediction performance. The checkpoint ensemble can have better selective prediction performance than the end checkpoint.

E.3.5 Empirical analysis for self-training

In this section, we analyze why the proposed self-training can improve selective prediction performance. Our hypothesis is that after fine-tuning the models on the selected labeled test data, the checkpoint ensemble constructed is less confident on

Dataset	DomainNet R→S (hard)								
Metric	cov|acc $\geqslant$ 70% ↑			acc|cov $\geqslant$ 70% ↑			AUC ↑		
Labeling Budget	500	1000	2000	500	1000	2000	500	1000	2000
SR+Uniform	12.12±0.7	12.42±0.4	15.88±0.2	27.01±0.6	27.74±0.3	31.29±0.3	41.12±0.8	41.89±0.2	46.17±0.3
SR+Confidence	11.06±1.1	11.48±0.5	14.49±1.5	26.53±1.4	27.98±0.2	31.31±0.7	40.26±1.6	41.65±0.2	45.46±1.1
SR+Entropy	10.91±0.3	12.45±0.6	14.65±0.6	26.84±0.5	28.72±0.5	31.07±0.6	40.47±0.6	42.61±0.8	45.31±0.4
SR+Margin	12.23±0.4	13.06±0.4	15.31±0.4	27.87±0.2	29.19±0.4	31.51±0.8	41.91±0.3	43.22±0.4	45.97±0.8
SR+kCG	9.03±0.2	9.76±0.2	11.41±0.2	23.32±0.4	24.06±0.4	25.68±0.4	36.63±0.3	37.57±0.4	39.80±0.3
SR+CLUE	12.39±0.3	14.17±1.0	15.80±0.8	27.82±0.4	29.68±0.4	30.62±0.8	42.00±0.4	44.19±0.7	45.58±0.9
SR+BADGE	12.18±0.9	13.13±1.0	15.83±0.7	27.68±1.0	28.96±0.7	32.00±0.4	41.72±1.1	43.28±0.9	46.60±0.6
DE+Uniform	15.91±0.5	17.55±0.4	21.33±0.3	31.37±0.5	32.57±0.4	36.12±0.2	46.28±0.5	47.79±0.4	51.64±0.2
DE+Entropy	13.70±0.3	16.31±0.5	19.58±0.4	30.38±0.4	32.45±0.2	36.18±0.2	44.79±0.5	47.15±0.2	50.87±0.3
DE+Confidence	13.73±0.2	16.21±0.2	19.22±0.4	30.55±0.3	33.02±0.1	36.29±0.5	45.05±0.3	47.59±0.0	50.84±0.4
DE+Margin	14.99±0.2	17.45±0.4	21.74±0.7	31.67±0.5	33.51±0.5	37.88±0.3	46.38±0.5	48.44±0.5	52.78±0.4
DE+Avg-KLD	15.75±0.5	18.14±0.7	22.15±0.3	31.36±0.2	33.79±0.2	37.96±0.2	46.29±0.1	48.77±0.3	53.02±0.3
DE+CLUE	14.76±0.5	17.38±0.1	19.75±0.4	31.05±0.4	32.58±0.2	34.61±0.4	45.80±0.3	47.74±0.1	50.09±0.2
DE+BADGE	14.97±0.1	17.49±0.3	21.71±0.3	31.35±0.2	33.46±0.1	37.35±0.3	46.03±0.1	48.31±0.1	52.33±0.2
ASPEST (ours)	**17.86**±0.4	**20.42**±0.4	**25.87**±0.4	**35.17**±0.1	**37.28**±0.3	**41.46**±0.2	**49.62**±0.1	**51.61**±0.4	**55.90**±0.2

Table E.8: Results of comparing ASPEST to the baselines on DomainNet R→S. The mean and std of each metric over three random runs are reported (mean±std). All numbers are percentages. **Bold** numbers are superior results.

the test data U_X compared to the deep ensemble (obtained by ensembling the end checkpoints). Thus, using the softmax outputs of the checkpoint ensemble as soft pseudo-labels for self-training can alleviate the overconfidence issue and improve selective prediction performance.

We perform experiments on the image dataset MNIST→SVHN and the text dataset Amazon Review to verity this hypothesis. To see the effect of self-training better, we only employ one-round active learning (i.e., only apply one-round self-training) with a labeling budget of 100 samples. We visualize the histogram of the confidence scores on the test data U_X for the deep ensemble and the checkpoint ensemble after fine-tuning. We also evaluate the receiver operator characteristic curve (AUROC) and the area under the accuracy-coverage curve (AUC) metrics of the checkpoint ensemble before and after the self-training. We use the AUROC metric to measure the quality of the confidence score for selective prediction. The results in Figure E.3 show that the checkpoint ensemble is less confident on the test data U_X compared to the deep ensemble. On the high-confidence region (i.e., confidence$\geqslant \eta$. Recall that η is the confidence threshold used for constructing the pseudo-labeled set R. We set $\eta = 0.9$ in our experiments), the checkpoint ensemble

Dataset	Otto								
Metric	cov|acc $\geqslant$ 80% $\uparrow$			acc|cov $\geqslant$ 80% $\uparrow$			AUC $\uparrow$		
Labeling Budget	500	1000	2000	500	1000	2000	500	1000	2000
SR+Uniform	63.58±0.7	64.06±0.4	67.49±0.9	73.56±0.3	73.57±0.6	75.21±0.2	84.46±0.2	84.61±0.3	85.72±0.2
SR+Confidence	69.63±1.7	73.41±0.6	84.19±0.5	75.96±0.5	77.57±0.2	81.39±0.2	85.91±0.3	86.86±0.1	88.93±0.1
SR+Entropy	67.79±0.8	73.83±1.0	83.12±0.7	75.43±0.4	77.91±0.3	81.07±0.2	85.41±0.3	86.94±0.2	88.86±0.1
SR+Margin	68.10±0.1	74.10±0.4	82.53±0.2	75.52±0.0	77.66±0.1	80.93±0.1	85.56±0.1	86.99±0.1	88.83±0.1
SR+kCG	64.84±0.7	62.90±1.1	59.85±1.0	73.75±0.3	73.03±0.2	71.90±0.3	85.08±0.2	84.67±0.2	83.79±0.3
SR+CLUE	68.21±1.2	70.85±0.6	78.26±0.9	75.26±0.5	76.32±0.2	79.30±0.3	85.82±0.3	86.69±0.2	88.53±0.2
SR+BADGE	67.23±1.0	73.52±0.2	83.17±0.4	74.74±0.3	77.43±0.2	81.20±0.2	85.41±0.3	87.10±0.2	89.25±0.1
DE+Uniform	70.74±0.5	72.20±0.6	75.58±0.5	76.40±0.1	77.06±0.2	78.35±0.2	86.78±0.1	87.26±0.1	88.11±0.1
DE+Entropy	75.71±0.3	80.91±0.2	92.62±0.2	78.44±0.1	80.29±0.1	84.05±0.1	87.87±0.1	88.77±0.1	90.99±0.1
DE+Confidence	75.52±0.2	81.69±0.7	92.15±0.9	78.28±0.1	80.49±0.2	83.83±0.1	87.84±0.1	89.05±0.1	90.98±0.1
DE+Margin	75.49±0.8	81.36±0.8	92.49±0.4	78.41±0.3	80.50±0.2	84.06±0.2	87.89±0.2	89.10±0.2	90.95±0.2
DE+Avg-KLD	75.91±0.2	80.97±0.5	91.94±0.8	78.50±0.1	80.33±0.2	83.80±0.2	87.89±0.0	89.06±0.1	90.98±0.1
DE+CLUE	69.66±0.5	70.52±0.1	70.17±0.4	76.09±0.3	76.32±0.1	76.31±0.2	86.67±0.1	87.11±0.0	87.06±0.1
DE+BADGE	73.23±0.2	77.89±0.6	86.32±0.5	77.33±0.1	79.21±0.3	82.32±0.2	87.55±0.1	88.75±0.1	90.58±0.0
ASPEST (ours)	**77.85**±0.2	**84.20**±0.6	**94.26**±0.6	**79.28**±0.1	**81.40**±0.1	**84.62**±0.1	**88.28**±0.1	**89.61**±0.1	**91.49**±0.0

Table E.9: Results of comparing ASPEST to the baselines on Otto. The mean and std of each metric over three random runs are reported (mean±std). All numbers are percentages. **Bold** numbers are superior results.

Dataset	CIFAR-10→CINIC-10		Otto	
Metric	cov*|acc $\geqslant$ 90% $\uparrow$	acc|cov* $\geqslant$ 90% $\uparrow$	cov*|acc $\geqslant$ 80% $\uparrow$	acc|cov* $\geqslant$ 80% $\uparrow$
SR (w/o active learning)	57.43±0.0	75.62±0.0	62.90±0.0	73.13±0.0
SR+Margin (M=500)	56.76±0.8	75.61±0.2	65.34±0.1	74.25±0.1
SR+Margin (M=1000)	56.04±0.7	75.70±0.1	68.11±0.4	74.99±0.2
DE (w/o active learning)	56.64±0.2	75.83±0.1	67.69±0.4	75.41±0.2
DE+Margin (M=500)	57.78±0.5	76.96±0.2	72.44±0.7	77.18±0.3
DE+Margin (M=1000)	59.55±0.5	77.59±0.1	74.78±0.7	78.19±0.2
ASPEST (M=500)	59.37±0.3	77.60±0.1	74.71±0.2	77.99±0.2
ASPEST (M=1000)	**61.23**±0.2	**78.16**±0.1	**77.40**±0.5	**79.05**±0.2

Table E.10: Results on CIFAR-10→CINIC-10 and Otto for studying the effect of combining selective prediction with active learning. "w/o" means "without". The mean and std of each metric over three random runs are reported (mean±std). All numbers are percentages. **Bold** numbers are superior results.

is also less confident than the deep ensemble. Besides, the results in Table E.13 show that after self-training, both AUROC and AUC metrics of the checkpoint ensemble are improved significantly. Therefore, the self-training can alleviate the overconfidence issue and improve selective prediction performance.

Dataset	FMoW		Amazon Review	
Metric	$\text{cov}^*\|\text{acc} \geqslant 70\% \uparrow$	$\text{acc}\|\text{cov}^* \geqslant 70\% \uparrow$	$\text{cov}^*\|\text{acc} \geqslant 80\% \uparrow$	$\text{acc}\|\text{cov}^* \geqslant 80\% \uparrow$
SR (w/o active learning)	32.39±0.0	48.15±0.0	26.79±0.0	65.64±0.0
SR+Margin (M=500)	37.54±1.3	52.19±0.3	14.16±10.6	65.38±0.4
SR+Margin (M=1000)	42.65±0.7	55.30±0.5	21.60±4.0	65.68±0.4
DE (w/o active learning)	37.58±0.3	52.01±0.1	35.81±1.9	68.41±0.2
DE+Margin (M=500)	45.30±0.6	57.09±0.3	32.68±1.2	68.10±0.3
DE+Margin (M=1000)	52.32±1.2	60.96±0.4	33.47±1.2	68.54±0.2
ASPEST (M=500)	51.85±0.4	60.43±0.2	37.59±0.6	68.91±0.2
ASPEST (M=1000)	**57.15**±0.4	**63.71**±0.2	**39.14**±0.8	**69.31**±0.2

Table E.11: Results on FMoW and Amazon Review for studying the effect of combining selective prediction with active learning. "w/o" means "without". The mean and std of each metric over three random runs are reported (mean±std). All numbers are percentages. **Bold** numbers are superior results.

Dataset	DomainNet R→C (easy)		DomainNet R→P (medium)		DomainNet R→S (hard)	
Metric	$\text{cov}^*\|\text{acc} \geqslant 80\% \uparrow$	$\text{acc}\|\text{cov}^* \geqslant 80\% \uparrow$	$\text{cov}^*\|\text{acc} \geqslant 70\% \uparrow$	$\text{acc}\|\text{cov}^* \geqslant 70\% \uparrow$	$\text{cov}^*\|\text{acc} \geqslant 70\% \uparrow$	$\text{acc}\|\text{cov}^* \geqslant 70\% \uparrow$
SR (w/o active learning)	21.50±0.0	40.16±0.0	18.16±0.0	34.74±0.0	7.16±0.0	21.24±0.0
SR+Margin (M=500)	25.38±1.1	44.09±0.9	20.94±0.4	36.65±0.1	11.94±0.4	27.35±0.2
SR+Margin (M=1000)	25.87±1.0	44.70±0.7	22.22±0.3	37.91±0.4	12.43±0.4	28.19±0.4
DE (w/o active learning)	26.15±0.2	44.51±0.1	22.44±0.2	39.06±0.1	9.90±0.4	25.37±0.0
DE+Margin (M=500)	30.73±1.2	48.85±0.4	25.19±0.3	40.59±0.1	14.63±0.2	31.11±0.5
DE+Margin (M=1000)	33.24±0.2	50.46±0.4	26.60±0.5	41.73±0.3	16.62±0.4	32.30±0.5
ASPEST (M=500)	36.10±0.1	53.22±0.3	29.01±0.1	44.26±0.1	17.43±0.4	34.55±0.1
ASPEST (M=1000)	**37.24**±0.3	**54.03**±0.1	**31.01**±0.3	**45.31**±0.1	**19.45**±0.3	**35.96**±0.3

Table E.12: Results on DomainNet R→C, R→P and R→S for studying the effect of combining selective prediction with active learning. "w/o" means "without". The mean and std of each metric over three random runs are reported (mean±std). All numbers are percentages. **Bold** numbers are superior results.

Dataset	MNIST→SVHN		Amazon Review	
Metric	AUROC↑	AUC↑	AUROC↑	AUC↑
Before self-training	73.92	66.75	67.44	76.24
After self-training	74.31	67.37	67.92	76.80

Table E.13: Evaluating the AUROC and AUC metrics of the checkpoint ensemble before and after self-training. All numbers are percentages.

E.3.6 Training with unsupervised domain adaptation

In this section, we study whether incorporating Unsupervised Domain Adaptation (UDA) techniques into training could improve the selective prediction performance. UDA techniques are mainly proposed to adapt the representation learned on the

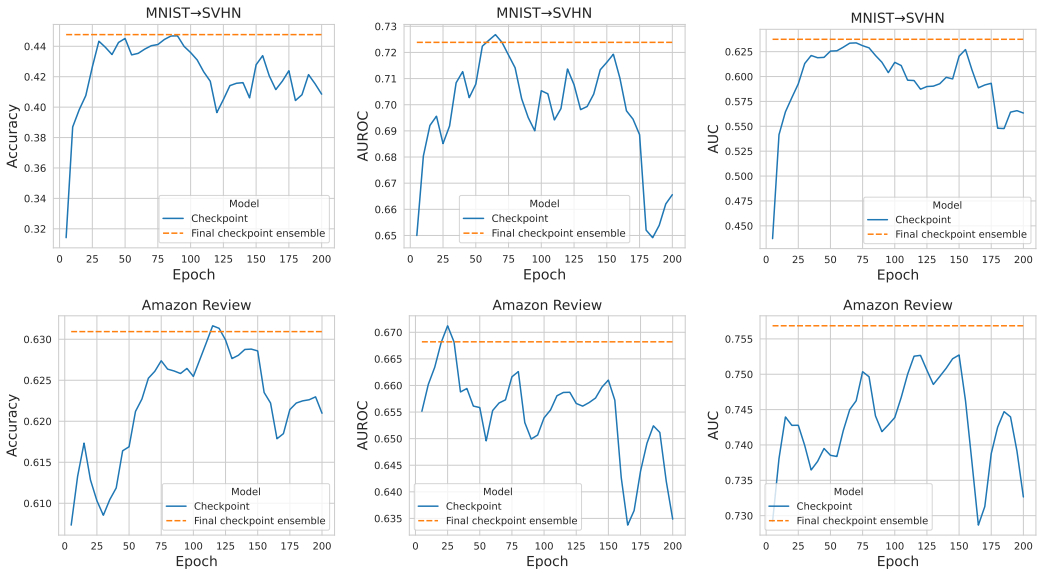


Figure E.2: Evaluating the checkpoints during fine-tuning and the checkpoint ensemble constructed after fine-tuning on the target test dataset.

labeled source domain data to the target domain with unlabeled data from the target domain (Liu et al., 2022). We can easily incorporate those UDA techniques into SR (Algorithm 5), DE (Algorithm 6), and the proposed ASPEST (Algorithm 4) by adding unsupervised training losses into the training objectives.

We consider the method `DE with UDA` and the method `ASPEST with UDA`. The algorithm for DE with UDA is presented in Algorithm 7 and the algorithm for ASPEST with UDA is presented in Algorithm 8. We consider UDA techniques based on representation matching where the goal is to minimize the distance between the distribution of the representation on $\mathcal{D}^{\text{tr}}$ and that on U_X. Suppose the model f is a composition of a prediction function h and a representation function ϕ (i.e., $f(x) = h(\phi(x))$). Then $L_{\text{UDA}}(\mathcal{D}^{\text{tr}}, U_X; \theta) = d(p^{\phi}_{\mathcal{D}^{\text{tr}}}, p^{\phi}_{U_X})$, which is a representation matching loss. We consider the representation matching losses from the state-of-the-art UDA methods DANN (Ganin et al., 2016) and CDAN (Long et al., 2018).

We evaluate two instantiations of Algorithm 7 – `DE with DANN` and `DE with CDAN`, and two instantiations of Algorithm 8 – `ASPEST with DANN` and `ASPEST with CDAN`. The values of the hyper-parameters are the same as those described in Sec-

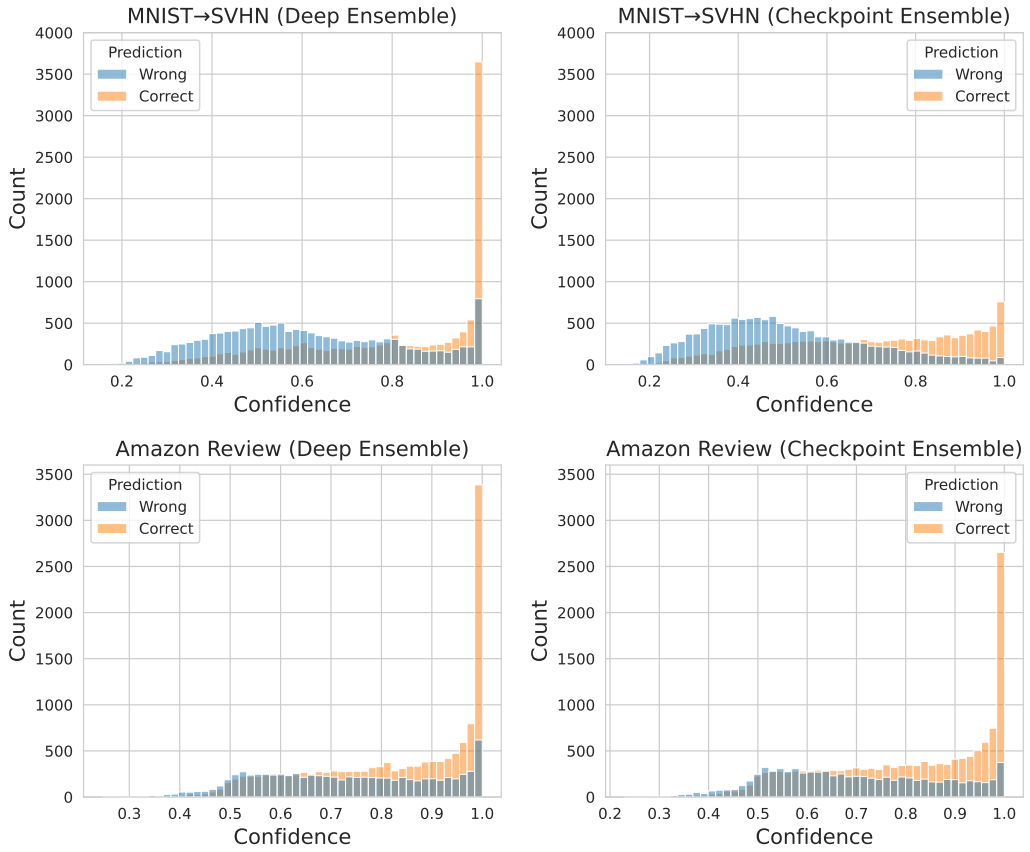


Figure E.3: Plot the histogram of the confidence scores on the test data U_X for the deep ensemble and the checkpoint ensemble after fine-tuning.

tion 7.5.1 except that we set $n_s = 20$. For DANN and CDAN, we set the hyper-parameter between the source classifier and the domain discriminator to be 0.1. The results are shown in Table E.14 (MNIST→SVHN), Table E.15 (CIFAR-10→CINIC-10), Table E.16 (FMoW), Table E.17 (Amazon Review), Table E.18 (DomainNet R→C), Table E.19 (DomainNet R→P), Table E.20 (DomainNet R→S) and Table E.21 (Otto).

From the results, we can see that ASPEST outperforms (or on par with) `DE with DANN` and `DE with CDAN` across different datasets, although ASPEST doesn't use UDA techniques. We further show that by combining ASPEST with UDA, it might achieve even better performance. For example, on MNIST→SVHN, `ASPEST with`

DANN improves the mean AUC from 96.62% to 97.03% when the labeling budget is 500. However, in some cases, combining ASPEST with DANN or CDAN leads to much worse results. For example, on MNIST→SVHN, when the labeling budget is 100, combining ASPEST with DANN or CDAN will reduce the mean AUC by over 4%. It might be because in those cases, DANN or CDAN fails to align the representations between the source and target domains. Existing work also show that UDA methods may not have stable performance across different kinds of distribution shifts and sometimes they can even yield accuracy degradation (Johansson et al., 2019; Sagawa et al., 2021). So our findings align with those of existing work.

Dataset	MNIST→SVHN								
Metric	cov|acc $\geqslant$ 90% ↑			acc|cov $\geqslant$ 90% ↑			AUC ↑		
Labeling Budget	100	500	1000	100	500	1000	100	500	1000
DE with DANN + Uniform	27.27±1.8	72.78±2.0	87.05±0.5	63.95±1.4	82.99±0.8	88.64±0.2	80.37±0.7	93.25±0.4	96.05±0.1
DE with DANN + Entropy	11.33±8.2	74.04±2.2	91.06±1.4	58.28±2.1	83.64±0.9	90.41±0.5	74.62±1.6	93.45±0.5	96.47±0.2
DE with DANN + Confidence	15.68±6.3	76.34±3.1	93.96±1.2	61.32±3.0	85.02±0.9	91.64±0.4	76.43±3.0	93.85±0.6	96.97±0.3
DE with DANN + Margin	30.64±2.1	83.44±0.9	96.17±0.5	66.79±0.9	87.30±0.4	92.71±0.2	82.14±0.8	95.40±0.3	97.60±0.1
DE with DANN + Avg-KLD	22.30±3.0	78.13±2.1	93.42±1.0	63.22±2.0	85.40±0.8	91.47±0.5	78.88±1.6	94.25±0.5	97.02±0.2
DE with DANN + CLUE	16.42±13.6	72.27±2.8	86.71±0.4	61.79±2.7	82.72±1.1	88.46±0.2	77.47±3.4	93.33±0.5	96.21±0.0
DE with DANN + BADGE	25.41±10.9	78.83±1.2	90.94±1.1	63.93±4.4	85.27±0.5	90.45±0.5	79.82±4.1	94.58±0.3	96.89±0.1
DE with CDAN + Uniform	28.10±4.8	73.15±0.7	87.50±0.6	63.95±2.7	83.10±0.3	88.86±0.3	80.28±2.2	93.44±0.1	96.13±0.2
DE with CDAN + Entropy	6.94±9.8	74.38±1.5	90.77±1.3	59.90±2.3	84.14±0.4	90.32±0.6	76.04±2.0	93.48±0.3	96.38±0.2
DE with CDAN + Confidence	13.47±10.2	75.15±2.8	92.77±0.7	60.98±2.0	84.62±0.9	91.23±0.3	76.19±2.8	93.62±0.6	96.63±0.1
DE with CDAN + Margin	22.44±3.3	81.84±2.5	96.07±0.2	62.89±3.8	86.71±1.0	92.64±0.0	78.69±2.6	94.89±0.5	97.57±0.0
DE with CDAN + Avg-KLD	20.23±4.1	80.62±1.7	93.13±2.5	62.23±2.7	86.34±0.6	91.30±1.0	77.68±2.5	94.81±0.4	96.97±0.4
DE with CDAN + CLUE	7.47±6.4	72.61±2.9	87.22±0.2	57.82±2.9	82.50±1.3	88.62±0.1	73.33±2.3	93.38±0.7	96.31±0.0
DE with CDAN + BADGE	26.88±3.5	79.21±0.1	92.50±0.7	65.69±1.7	85.32±0.1	91.18±0.4	81.10±1.3	94.73±0.1	97.17±0.2
ASPEST (ours)	**52.10**±4.0	89.22±0.9	98.70±0.4	**76.10**±1.5	89.62±0.4	93.92±0.3	**88.84**±1.0	96.62±0.2	98.06±0.1
ASPEST with DANN (ours)	37.90±2.4	**91.61**±0.6	99.39±0.4	69.45±1.7	**90.70**±0.3	94.42±0.4	84.55±1.0	**97.03**±0.1	98.23±0.1
ASPEST with CDAN (ours)	30.97±11.7	91.39±0.6	**99.50**±0.3	67.58±3.2	90.60±0.3	**94.46**±0.2	82.20±3.3	96.95±0.1	**98.26**±0.1

Table E.14: Results of evaluating DE with UDA and ASPEST with UDA on MNIST→SVHN. The mean and std of each metric over three random runs are reported (mean±std). All numbers are percentages. **Bold** numbers are superior results.

Algorithm 7 DE with Unsupervised Domain Adaptation

Require: A training dataset $\mathcal{D}^{\text{tr}}$, An unlabeled test dataset U_X, the number of rounds T, the total labeling budget M, a source-trained model $\bar{f}$, an acquisition function $a(B, f, g)$, the number of models in the ensemble N, the number of initial training epochs n_s, and a hyper-parameter λ.

Let $f_0^j = \bar{f}$ for $j = 1, \ldots, N$.

Fine-tune each model f_0^j in the ensemble via SGD for n_s training epochs independently using the following training objective with different randomness:

$$\min_{\theta^j} \quad \mathbb{E}_{(x,y)\in\mathcal{D}^{\text{tr}}} \quad \ell_{CE}(\mathbf{x}, y; \theta^j) + L_{UDA}(\mathcal{D}^{\text{tr}}, U_X; \theta^j) \tag{E.23}$$

where L_{UDA} is a loss function for unsupervised domain adaptation.

Let $B_0 = \emptyset$.

Let $g_t(\mathbf{x}) = \max_{k\in\mathcal{Y}} f_t(\mathbf{x} \mid k)$.

for $t = 1, \cdots, T$ **do**

Select a batch B_t with a size of $m = [\frac{M}{T}]$ from U_X for labeling via:

$$B_t = \arg \max_{B\subset U_X\setminus(\cup_{l=0}^{t-1} B_l), |B|=m} a(B, f_{t-1}, g_{t-1}) \tag{E.24}$$

Use an oracle to assign ground-truth labels to the examples in B_t to get $\tilde{B}_t$.

Fine-tune each model f_{t-1}^j in the ensemble via SGD independently using the following training objective with different randomness:

$$\min_{\theta^j} \quad \mathbb{E}_{(x,y)\in\cup_{l=1}^{t}\tilde{B}_l} \ell_{CE}(\mathbf{x}, y; \theta^j) + \lambda \cdot \mathbb{E}_{(x,y)\in\mathcal{D}^{\text{tr}}} \ell_{CE}(\mathbf{x}, y; \theta^j) + L_{UDA}(\mathcal{D}^{\text{tr}}, U_X; \theta^j) \tag{E.25}$$

where θ^j is the model parameters of f_{t-1}^j.

Let $f_t^j = f_{t-1}^j$.

end for

Ensure: The classifier $f = f_T$ and the selection scoring function $g = \max_{k\in\mathcal{Y}} f(\mathbf{x} \mid k)$.

Algorithm 8 ASPEST with Unsupervised Domain Adaptation

Require: A training set $\mathcal{D}^{\text{tr}}$, a unlabeled test set U_X, the number of rounds T, the labeling budget M, the number of models N, the number of initial training epochs n_s, a checkpoint epoch c_e, a threshold η, a sub-sampling fraction p, and a hyper-parameter λ.

Let $f_0^j = \bar{f}$ for $j = 1, \ldots, N$.

Set $N_e = 0$ and $P = \mathbf{0}_{n \times K}$.

Fine-tune each f_0^j for n_s training epochs using the following training objective:

$$\min_{\theta^j} \quad \mathbb{E}_{(\mathbf{x},y)\in\mathcal{D}^{\text{tr}}} \quad \ell_{CE}(\mathbf{x}, y; \theta^j) + L_{UDA}(\mathcal{D}^{\text{tr}}, U_X; \theta^j), \tag{E.26}$$

where L_{UDA} is a loss function for unsupervised domain adaptation. During fine-tuning, update P and N_e using Eq. (7.9) every c_e training epochs.

for $t = 1, \cdots, T$ **do**

Select a batch B_t from U_X for labeling using the sample selection objective (7.11).

Use an oracle to assign ground-truth labels to the examples in B_t to get $\tilde{B}_t$.

Set $N_e = 0$ and $P = \mathbf{0}_{n \times K}$.

Fine-tune each f_{t-1}^j using the following training objective:

$$\min_{\theta^j} \quad \mathbb{E}_{(\mathbf{x},y)\in\cup_{l=1}^{t}\tilde{B}_l} \quad \ell_{CE}(\mathbf{x}, y; \theta^j) + \lambda \cdot \mathbb{E}_{(\mathbf{x},y)\in\mathcal{D}^{\text{tr}}} \quad \ell_{CE}(\mathbf{x}, y; \theta^j) + L_{UDA}(\mathcal{D}^{\text{tr}}, U_X; \theta^j), \tag{E.27}$$

During fine-tuning, update P and N_e using Eq (7.9) every c_e training epochs.

Let $f_t^j = f_{t-1}^j$.

Construct the pseudo-labeled set R via Eq (7.13) and create R_{sub} by randomly sampling up to $\lceil p \cdot n \rceil$ data points from R.

Train each f_t^j further via SGD using the objective (7.14) and update P and N_e using Eq (7.9) every c_e training epochs.

end for

Ensure: The classifier $f(\mathbf{x}_i) = \arg\max_{k\in\mathcal{Y}} P_{i,k}$ and the selection scoring function $g(\mathbf{x}_i) = \max_{k\in\mathcal{Y}} P_{i,k}$.

Dataset	CIFAR-10→CINIC-10								
Metric	cov|acc $\geqslant$ 90% ↑			acc|cov $\geqslant$ 90% ↑			AUC ↑		
Labeling Budget	500	1000	2000	500	1000	2000	500	1000	2000
DE with DANN + Uniform	58.85±0.3	59.39±0.2	60.04±0.1	77.06±0.2	77.33±0.2	77.84±0.1	90.40±0.1	90.60±0.1	90.73±0.1
DE with DANN + Entropy	59.42±0.4	60.86±0.3	64.52±0.3	78.14±0.2	79.20±0.1	81.31±0.1	90.72±0.0	91.06±0.1	92.02±0.0
DE with DANN + Confidence	59.44±0.6	61.08±0.3	65.12±0.2	78.19±0.1	79.38±0.0	81.29±0.1	90.73±0.1	91.26±0.1	92.06±0.0
DE with DANN + Margin	59.81±0.3	62.26±0.4	65.58±0.4	78.15±0.0	79.25±0.1	81.05±0.1	90.76±0.1	91.30±0.1	92.11±0.0
DE with DANN + Avg-KLD	60.50±0.5	62.04±0.1	65.08±0.2	78.32±0.1	79.31±0.1	81.07±0.0	90.89±0.1	91.34±0.0	92.11±0.0
DE with DANN + CLUE	60.20±0.5	61.69±0.2	64.08±0.2	77.84±0.2	78.35±0.2	79.38±0.1	90.73±0.2	91.07±0.1	91.63±0.0
DE with DANN + BADGE	60.18±0.4	62.15±0.2	65.31±0.6	77.70±0.1	78.54±0.1	79.81±0.2	90.72±0.1	91.19±0.1	91.86±0.1
DE with CDAN + Uniform	58.72±0.2	59.49±0.5	60.28±0.2	77.16±0.0	77.52±0.1	77.90±0.1	90.45±0.1	90.65±0.0	90.78±0.1
DE with CDAN + Entropy	58.73±0.4	60.82±0.5	64.45±0.2	77.95±0.1	79.20±0.1	81.04±0.1	90.57±0.1	91.10±0.1	91.86±0.1
DE with CDAN + Confidence	59.10±0.6	61.03±0.6	64.60±0.2	77.92±0.0	79.26±0.2	81.07±0.0	90.59±0.0	91.10±0.2	91.96±0.0
DE with CDAN + Margin	59.88±0.5	61.57±0.9	64.82±0.4	78.09±0.3	79.02±0.2	80.82±0.1	90.73±0.1	91.17±0.2	91.98±0.1
DE with CDAN + Avg-KLD	60.51±0.1	61.71±0.5	65.03±0.3	78.20±0.2	79.29±0.2	81.15±0.1	90.85±0.0	91.19±0.1	92.07±0.1
DE with CDAN + CLUE	60.12±0.5	61.77±0.3	64.06±0.2	77.88±0.1	78.38±0.2	79.42±0.2	90.73±0.1	91.08±0.1	91.64±0.0
DE with CDAN + BADGE	60.28±0.7	61.84±0.2	65.29±0.3	77.68±0.2	78.53±0.1	79.84±0.2	90.73±0.1	91.17±0.0	91.95±0.1
ASPEST (ours)	60.38±0.3	63.34±0.2	66.81±0.3	78.23±0.1	79.49±0.1	81.25±0.1	90.95±0.0	91.60±0.0	92.33±0.1
ASPEST with DANN (ours)	**61.69**±0.2	**63.58**±0.4	**66.81**±0.4	**78.68**±0.1	**79.68**±0.1	81.42±0.1	**91.16**±0.1	**91.66**±0.1	92.37±0.1
ASPEST with CDAN (ours)	61.00±0.2	62.80±0.4	66.78±0.1	78.56±0.1	79.54±0.1	**81.49**±0.0	91.13±0.0	91.57±0.1	**92.41**±0.0

Table E.15: Results of evaluating DE with UDA and ASPEST with UDA on CIFAR-10→CINIC-10. The mean and std of each metric over three random runs are reported (mean±std). All numbers are percentages. **Bold** numbers are superior results.

Dataset	FMoW								
Metric	cov|acc $\geqslant$ 70% ↑			acc|cov $\geqslant$ 70% ↑			AUC ↑		
Labeling Budget	500	1000	2000	500	1000	2000	500	1000	2000
DE with DANN + Uniform	46.11±0.6	51.77±0.3	62.76±0.5	57.62±0.3	60.67±0.4	66.21±0.2	70.17±0.3	72.46±0.3	76.83±0.2
DE with DANN + Entropy	44.36±0.7	48.19±0.3	59.52±0.8	56.78±0.1	59.51±0.0	65.75±0.3	69.09±0.2	71.02±0.2	75.15±0.3
DE with DANN + Confidence	44.46±0.5	49.32±0.1	61.47±0.3	57.04±0.3	60.51±0.3	66.61±0.1	69.14±0.1	71.50±0.1	75.70±0.1
DE with DANN + Margin	48.09±0.4	54.35±0.5	70.11±0.4	59.07±0.2	62.79±0.2	70.02±0.1	70.76±0.1	73.29±0.2	78.25±0.1
DE with DANN + Avg-KLD	48.42±0.1	55.95±0.2	68.73±1.1	59.06±0.2	63.44±0.2	69.41±0.5	70.84±0.1	73.83±0.1	77.91±0.4
DE with DANN + CLUE	44.14±0.6	46.15±0.2	49.02±0.5	56.01±0.3	56.89±0.2	58.66±0.3	69.11±0.2	70.16±0.2	71.46±0.2
DE with DANN + BADGE	48.57±0.5	54.47±0.5	67.69±0.9	58.61±0.2	61.67±0.0	68.71±0.5	71.17±0.2	73.64±0.1	78.65±0.3
DE with CDAN + Uniform	46.08±0.7	51.92±0.8	62.87±0.2	57.45±0.1	60.73±0.4	66.19±0.2	69.93±0.3	72.57±0.4	76.87±0.1
DE with CDAN + Entropy	44.42±0.3	49.32±0.1	60.11±0.3	56.83±0.1	60.04±0.2	65.95±0.2	69.18±0.2	71.34±0.3	75.44±0.3
DE with CDAN + Confidence	44.75±0.1	49.34±0.1	62.80±1.0	57.09±0.1	60.50±0.2	66.94±0.4	69.27±0.1	71.60±0.2	76.14±0.3
DE with CDAN + Margin	47.48±0.7	54.48±0.7	70.25±0.9	58.98±0.4	62.98±0.3	70.10±0.4	70.55±0.3	73.46±0.2	78.39±0.3
DE with CDAN + Avg-KLD	48.43±0.2	54.37±0.4	68.93±0.6	59.36±0.2	62.71±0.2	69.54±0.2	71.12±0.2	73.35±0.2	77.97±0.2
DE with CDAN + CLUE	44.09±0.3	46.11±0.5	48.90±0.1	55.78±0.3	56.98±0.2	58.46±0.2	69.03±0.1	70.02±0.2	71.31±0.1
DE with CDAN + BADGE	47.93±0.2	54.61±0.2	67.01±0.5	58.16±0.1	61.81±0.1	68.36±0.2	70.91±0.2	73.63±0.1	78.52±0.2
ASPEST (ours)	**53.05**±0.4	**59.86**±0.4	**76.52**±0.6	61.18±0.2	**65.18**±0.2	**72.86**±0.3	71.12±0.2	**74.25**±0.2	**79.93**±0.1
ASPEST with DANN (ours)	51.02±0.9	58.63±1.1	72.97±0.9	61.10±0.5	64.98±0.4	71.21±0.4	71.03±0.3	73.79±0.4	77.84±0.3
ASPEST with CDAN (ours)	51.40±0.6	58.21±0.6	73.94±0.6	**61.38**±0.2	65.04±0.2	71.63±0.2	**71.17**±0.1	73.59±0.1	78.04±0.2

Table E.16: Results of evaluating DE with UDA and ASPEST with UDA on FMoW. The mean and std of each metric over three random runs are reported (mean±std). All numbers are percentages. **Bold** numbers are superior results.

Dataset	Amazon Review								
Metric	cov|acc $\geqslant$ 80% ↑			acc|cov $\geqslant$ 80% ↑			AUC ↑		
Labeling Budget	500	1000	2000	500	1000	2000	500	1000	2000
DE with DANN + Uniform	38.55±3.3	37.25±1.8	39.21±1.9	69.06±0.6	68.94±0.1	69.41±0.2	77.52±0.7	77.03±0.4	77.70±0.2
DE with DANN + Entropy	38.22±2.3	41.85±0.8	41.57±1.3	69.48±0.3	**70.71**±0.3	71.55±0.2	77.49±0.5	78.39±0.2	78.58±0.1
DE with DANN + Confidence	38.01±1.0	38.36±2.5	38.89±1.3	69.45±0.1	70.16±0.3	71.44±0.2	77.54±0.2	77.58±0.5	78.48±0.3
DE with DANN + Margin	36.82±1.3	36.89±1.3	41.98±1.5	69.35±0.3	69.63±0.3	71.27±0.2	77.30±0.3	77.23±0.3	78.34±0.3
DE with DANN + Avg-KLD	37.15±2.9	38.21±1.3	42.46±1.4	69.38±0.4	69.79±0.2	71.21±0.2	77.25±0.6	77.72±0.3	78.68±0.3
DE with DANN + CLUE	**40.23**±4.0	34.71±1.8	31.38±0.9	68.95±0.7	68.07±0.2	67.44±0.3	77.62±1.0	76.27±0.6	75.60±0.2
DE with DANN + BADGE	37.51±1.8	37.00±0.9	41.62±2.3	68.98±0.4	69.27±0.1	70.20±0.4	77.20±0.4	77.21±0.1	78.31±0.5
DE with CDAN + Uniform	37.81±0.3	37.83±2.7	39.52±0.8	68.93±0.1	69.16±0.7	69.50±0.3	77.16±0.1	77.30±0.7	77.74±0.3
DE with CDAN + Entropy	37.99±0.8	37.68±1.1	42.55±0.9	**69.54**±0.3	70.01±0.2	71.52±0.2	77.52±0.2	77.61±0.1	78.63±0.1
DE with CDAN + Confidence	35.76±0.9	38.69±2.8	41.43±2.1	69.24±0.0	70.45±0.4	71.50±0.4	77.08±0.2	77.82±0.4	78.47±0.3
DE with CDAN + Margin	37.68±2.9	37.43±1.0	42.18±1.3	69.50±0.3	69.80±0.4	71.29±0.0	77.50±0.5	77.31±0.3	78.46±0.3
DE with CDAN + Avg-KLD	37.85±1.6	40.71±0.9	44.35±0.9	69.41±0.3	70.29±0.1	71.28±0.2	77.28±0.5	78.11±0.2	78.86±0.2
DE with CDAN + CLUE	34.85±2.7	34.03±1.3	30.70±0.4	68.70±0.3	67.84±0.1	67.12±0.3	76.95±0.7	76.23±0.4	75.36±0.4
DE with CDAN + BADGE	39.47±0.2	39.29±1.1	41.64±0.9	69.33±0.0	69.34±0.2	70.58±0.2	77.52±0.2	77.49±0.2	78.24±0.3
ASPEST (ours)	38.44±0.7	40.96±0.8	45.77±0.1	69.31±0.3	70.17±0.2	**71.60**±0.2	77.69±0.1	78.35±0.2	**79.51**±0.2
ASPEST with DANN (ours)	40.22±0.5	41.99±1.4	**45.84**±0.1	69.42±0.1	70.30±0.1	71.58±0.2	**78.00**±0.1	78.34±0.3	79.43±0.1
ASPEST with CDAN (ours)	40.02±0.5	**42.46**±0.6	44.95±0.4	69.50±0.1	70.37±0.2	71.42±0.0	77.80±0.1	**78.57**±0.1	79.25±0.0

Table E.17: Results of evaluating DE with UDA and ASPEST with UDA on Amazon Review. The mean and std of each metric over three random runs are reported (mean±std). All numbers are percentages. **Bold** numbers are superior results.

Dataset	DomainNet R→C (easy)								
Metric	cov|acc $\geqslant$ 80% ↑			acc|cov $\geqslant$ 80% ↑			AUC ↑		
Labeling Budget	500	1000	2000	500	1000	2000	500	1000	2000
DE with DANN + Uniform	33.53±0.5	36.28±0.3	40.13±1.0	50.57±0.5	52.19±0.1	55.15±0.1	69.34±0.3	70.98±0.2	73.50±0.3
DE with DANN + Entropy	28.66±1.0	34.47±0.1	42.77±0.7	48.13±0.6	52.70±0.3	59.01±0.2	66.60±0.5	70.64±0.1	75.45±0.2
DE with DANN + Confidence	29.92±0.4	35.29±1.0	43.33±0.4	48.61±0.1	53.36±0.5	59.72±0.3	67.23±0.2	70.92±0.5	75.89±0.3
DE with DANN + Margin	35.19±0.3	39.63±0.2	46.51±0.5	52.29±0.3	55.60±0.2	60.97±0.4	70.70±0.1	73.41±0.1	77.24±0.3
DE with DANN + Avg-KLD	36.02±0.6	39.67±0.5	47.20±0.8	53.00±0.3	55.75±0.3	61.22±0.3	71.19±0.3	73.51±0.2	77.46±0.2
DE with DANN + CLUE	32.26±1.5	35.09±0.4	35.66±0.3	50.21±0.0	50.90±0.1	51.50±0.1	69.17±0.2	70.20±0.2	70.82±0.1
DE with DANN + BADGE	35.27±0.5	38.88±0.3	45.97±0.7	52.15±0.3	54.89±0.1	60.03±0.3	70.65±0.1	72.95±0.1	76.87±0.1
DE with CDAN + Uniform	33.49±0.6	36.01±0.7	39.93±0.2	50.46±0.2	51.89±0.1	55.23±0.2	69.32±0.3	70.86±0.3	73.55±0.2
DE with CDAN + Entropy	29.50±0.5	33.86±0.3	42.24±0.5	48.01±0.1	52.52±0.3	58.96±0.2	66.82±0.2	70.28±0.1	75.33±0.1
DE with CDAN + Confidence	29.21±1.0	34.92±0.6	43.36±0.4	48.48±0.4	52.85±0.4	59.88±0.4	66.82±0.5	70.61±0.4	75.93±0.3
DE with CDAN + Margin	35.87±0.7	38.37±0.4	46.42±0.6	52.58±0.1	55.28±0.2	61.20±0.2	70.95±0.2	72.95±0.2	77.26±0.1
DE with CDAN + Avg-KLD	36.21±0.6	40.08±0.3	47.62±0.4	52.95±0.3	55.93±0.1	61.56±0.2	71.29±0.3	73.60±0.1	77.58±0.2
DE with CDAN + CLUE	31.74±2.1	35.11±0.2	35.87±0.5	49.99±0.2	51.39±0.2	51.43±0.2	69.04±0.3	70.35±0.0	70.82±0.3
DE with CDAN + BADGE	34.74±0.5	38.68±0.7	45.87±1.0	51.80±0.3	54.75±0.2	60.22±0.1	70.38±0.1	72.90±0.2	76.85±0.2
ASPEST (ours)	37.38±0.1	39.98±0.3	48.29±1.0	54.56±0.3	56.95±0.1	62.69±0.2	71.61±0.2	73.27±0.2	77.40±0.4
ASPEST with DANN (ours)	**37.41**±0.8	42.45±1.0	49.74±0.6	**55.60**±0.1	58.29±0.2	63.64±0.2	71.88±0.2	74.18±0.4	78.09±0.0
ASPEST with CDAN (ours)	36.60±1.2	**42.96**±0.6	**50.86**±0.2	55.55±0.2	**58.71**±0.2	**63.85**±0.2	**71.99**±0.2	**74.60**±0.2	**78.45**±0.3

Table E.18: Results of evaluating DE with UDA and ASPEST with UDA on DomainNet R→C. The mean and std of each metric over three random runs are reported (mean±std). All numbers are percentages. **Bold** numbers are superior results.

Dataset	DomainNet R→P (medium)								
Metric	cov|acc $\geqslant$ 70% ↑			acc|cov $\geqslant$ 70% ↑			AUC ↑		
Labeling Budget	500	1000	2000	500	1000	2000	500	1000	2000
DE with DANN + Uniform	26.98±0.1	28.34±0.5	30.63±0.2	41.96±0.2	42.89±0.2	44.73±0.1	57.04±0.1	58.10±0.2	59.87±0.1
DE with DANN + Entropy	24.75±0.4	27.02±0.5	30.10±0.2	40.29±0.4	42.34±0.2	45.78±0.2	55.19±0.3	57.12±0.3	60.21±0.1
DE with DANN + Confidence	22.41±0.9	27.03±0.6	31.70±0.6	39.05±0.5	42.61±0.2	46.60±0.2	53.66±0.6	57.35±0.3	60.93±0.4
DE with DANN + Margin	29.16±0.1	30.58±0.3	33.64±0.6	43.78±0.2	45.17±0.2	47.69±0.4	58.76±0.1	59.94±0.0	62.19±0.4
DE with DANN + Avg-KLD	29.52±0.1	31.17±0.4	34.09±0.3	43.84±0.3	45.33±0.2	48.18±0.2	58.89±0.2	60.25±0.2	62.54±0.2
DE with DANN + CLUE	27.48±0.5	27.83±0.2	28.39±0.5	42.05±0.3	42.34±0.2	42.65±0.1	57.32±0.3	57.64±0.2	57.99±0.2
DE with DANN + BADGE	28.92±0.1	30.36±0.2	33.86±0.3	43.38±0.1	44.85±0.1	47.64±0.3	58.38±0.0	59.82±0.1	62.26±0.2
DE with CDAN + Uniform	26.96±0.4	28.33±0.2	29.98±0.4	41.77±0.3	42.85±0.2	44.23±0.4	56.86±0.4	58.01±0.0	59.42±0.4
DE with CDAN + Entropy	24.91±0.4	26.30±0.9	30.33±0.4	40.34±0.3	42.07±0.6	45.79±0.2	55.38±0.4	56.70±0.8	60.23±0.2
DE with CDAN + Confidence	24.58±0.7	27.11±0.5	31.07±0.5	40.32±0.2	42.64±0.3	46.25±0.3	55.14±0.3	57.40±0.3	60.63±0.3
DE with CDAN + Margin	28.33±0.1	30.17±0.3	33.54±0.4	43.44±0.4	44.77±0.1	47.56±0.2	58.31±0.2	59.65±0.1	62.17±0.2
DE with CDAN + Avg-KLD	28.69±0.2	30.99±0.9	34.30±0.2	43.64±0.2	45.34±0.2	48.22±0.1	58.60±0.1	60.15±0.4	62.67±0.1
DE with CDAN + CLUE	27.52±0.6	27.96±0.2	28.18±0.5	42.02±0.2	42.44±0.1	42.67±0.2	57.21±0.3	57.70±0.1	58.04±0.3
DE with CDAN + BADGE	28.79±0.1	30.28±0.1	33.77±0.4	43.45±0.0	44.73±0.3	47.84±0.2	58.47±0.1	59.64±0.2	62.37±0.2
ASPEST (ours)	29.69±0.1	32.50±0.3	35.46±0.6	44.96±0.1	46.77±0.2	49.42±0.1	58.74±0.0	60.36±0.0	62.84±0.2
ASPEST with DANN (ours)	**31.75**±0.4	**33.58**±0.3	36.96±0.2	**46.16**±0.1	47.64±0.2	**50.37**±0.3	**59.63**±0.2	61.06±0.1	**63.75**±0.1
ASPEST with CDAN (ours)	30.39±0.4	33.57±0.3	**37.53**±0.7	45.90±0.1	**47.71**±0.2	50.31±0.2	59.13±0.3	**61.17**±0.2	63.69±0.3

Table E.19: Results of evaluating DE with UDA and ASPEST with UDA on DomainNet R→P. The mean and std of each metric over three random runs are reported (mean±std). All numbers are percentages. **Bold** numbers are superior results.

Dataset	DomainNet R→S (hard)								
Metric	cov|acc $\geqslant$ 70% ↑			acc|cov $\geqslant$ 70% ↑			AUC ↑		
Labeling Budget	500	1000	2000	500	1000	2000	500	1000	2000
DE with DANN + Uniform	17.55±0.4	19.82±0.3	23.57±0.4	32.61±0.5	34.56±0.3	37.73±0.2	47.60±0.5	49.92±0.4	53.52±0.1
DE with DANN + Entropy	10.77±0.8	15.38±0.5	20.11±0.5	27.78±0.7	31.09±0.2	36.39±0.3	41.69±0.7	45.62±0.3	51.05±0.4
DE with DANN + Confidence	10.64±1.2	15.22±0.4	20.25±0.5	28.09±1.0	31.76±0.3	36.86±0.8	41.94±1.3	46.19±0.3	51.48±0.7
DE with DANN + Margin	17.90±0.7	20.44±0.6	25.52±0.4	33.61±0.1	35.79±0.5	40.29±0.3	48.67±0.1	51.03±0.6	55.64±0.4
DE with DANN + Avg-KLD	18.02±1.0	21.22±0.2	25.46±0.2	34.00±0.2	36.51±0.2	40.72±0.2	49.05±0.2	51.79±0.2	55.95±0.2
DE with DANN + CLUE	15.77±0.3	18.14±0.7	19.49±0.4	32.10±0.1	33.42±0.3	34.50±0.3	47.18±0.2	48.63±0.3	50.03±0.3
DE with DANN + BADGE	16.84±0.9	20.88±0.3	25.11±0.3	33.97±0.1	36.20±0.2	40.01±0.3	48.87±0.2	51.46±0.2	55.33±0.2
DE with CDAN + Uniform	17.33±0.5	19.79±0.1	22.99±0.5	32.47±0.5	34.59±0.3	37.88±0.2	47.49±0.5	50.02±0.2	53.51±0.3
DE with CDAN + Entropy	12.48±0.8	15.19±0.8	20.23±0.0	28.83±0.1	32.41±0.4	36.57±0.1	42.93±0.5	47.00±0.3	51.24±0.2
DE with CDAN + Confidence	11.23±0.6	13.93±0.1	18.45±1.3	28.67±0.3	31.35±0.4	35.56±0.8	42.87±0.5	45.40±0.7	49.80±1.0
DE with CDAN + Margin	18.06±0.7	20.39±0.3	25.05±0.3	33.98±0.2	35.76±0.2	40.11±0.1	49.15±0.1	50.92±0.1	55.27±0.1
DE with CDAN + Avg-KLD	18.63±1.0	20.80±0.3	25.49±0.9	34.19±0.4	36.41±0.2	40.53±0.5	49.45±0.5	51.58±0.1	55.74±0.5
DE with CDAN + CLUE	16.51±0.3	18.82±0.1	19.47±0.1	32.23±0.2	33.83±0.4	34.72±0.3	47.40±0.2	49.11±0.2	49.98±0.3
DE with CDAN + BADGE	17.52±0.8	21.48±0.5	25.35±0.4	33.53±0.5	36.19±0.4	40.31±0.3	48.67±0.5	51.65±0.3	55.62±0.3
ASPEST (ours)	17.86±0.4	20.42±0.4	25.87±0.4	35.17±0.1	37.28±0.3	41.46±0.2	49.62±0.1	51.61±0.4	55.90±0.2
ASPEST with DANN (ours)	16.35±1.2	**23.18**±0.4	28.00±0.1	36.56±0.2	**39.40**±0.4	42.94±0.1	50.58±0.4	**53.73**±0.3	57.25±0.1
ASPEST with CDAN (ours)	**18.81**±1.1	22.95±0.8	**28.17**±0.2	**36.85**±0.3	39.10±0.2	**43.25**±0.3	**51.14**±0.3	53.47±0.2	**57.26**±0.2

Table E.20: Results of evaluating DE with UDA and ASPEST with UDA on DomainNet R→S. The mean and std of each metric over three random runs are reported (mean±std). All numbers are percentages. **Bold** numbers are superior results.

Dataset	Otto								
Metric	cov|acc $\geqslant$ 80% ↑			acc|cov $\geqslant$ 80% ↑			AUC ↑		
Labeling Budget	500	1000	2000	500	1000	2000	500	1000	2000
DE with DANN + Uniform	70.35±0.5	72.42±0.4	75.63±0.7	76.12±0.3	77.04±0.1	78.25±0.1	86.67±0.1	87.16±0.1	88.09±0.1
DE with DANN + Entropy	75.27±0.3	81.25±0.1	92.23±0.3	78.14±0.1	80.45±0.0	83.73±0.1	87.73±0.1	88.91±0.0	90.90±0.1
DE with DANN + Confidence	74.66±0.3	81.62±0.1	92.57±0.6	78.05±0.2	80.50±0.0	83.67±0.2	87.51±0.1	89.06±0.1	90.94±0.1
DE with DANN + Margin	75.47±0.4	82.56±0.7	91.86±0.9	78.26±0.1	80.79±0.2	83.61±0.3	87.87±0.1	89.08±0.0	90.88±0.1
DE with DANN + Avg-KLD	76.02±0.6	81.78±0.4	91.82±0.3	78.53±0.0	80.70±0.1	83.88±0.0	87.99±0.0	89.17±0.0	90.90±0.1
DE with DANN + CLUE	69.68±0.4	68.07±0.3	62.70±0.6	75.81±0.3	75.44±0.0	73.49±0.3	86.68±0.2	86.31±0.1	84.89±0.2
DE with DANN + BADGE	74.69±0.5	79.04±0.6	87.63±0.4	77.97±0.1	79.57±0.3	82.99±0.1	87.82±0.1	88.92±0.1	90.67±0.1
DE with CDAN + Uniform	70.25±0.9	72.43±0.4	75.21±0.7	76.09±0.3	76.94±0.1	78.13±0.1	86.56±0.3	87.14±0.2	87.90±0.1
DE with CDAN + Entropy	74.73±0.6	81.60±0.8	92.58±0.2	77.97±0.2	80.59±0.3	83.81±0.2	87.47±0.1	88.93±0.1	90.84±0.1
DE with CDAN + Confidence	74.88±0.6	81.30±0.8	92.53±0.9	78.06±0.2	80.51±0.3	83.85±0.3	87.43±0.2	88.99±0.1	90.95±0.1
DE with CDAN + Margin	76.68±1.0	81.57±0.4	92.20±0.5	78.74±0.5	80.62±0.2	84.01±0.2	88.08±0.2	88.85±0.2	91.09±0.0
DE with CDAN + Avg-KLD	75.88±0.4	81.82±0.8	91.43±1.1	78.45±0.1	80.72±0.3	83.72±0.3	87.92±0.2	89.12±0.2	90.91±0.2
DE with CDAN + CLUE	69.86±0.5	67.79±0.2	63.46±0.9	76.09±0.2	75.42±0.3	73.66±0.3	86.81±0.1	86.25±0.1	85.00±0.1
DE with CDAN + BADGE	74.68±0.4	79.46±0.3	87.57±0.4	77.89±0.1	79.78±0.1	82.85±0.1	87.78±0.1	88.90±0.1	90.72±0.1
ASPEST (ours)	77.85±0.2	**84.20**±0.6	94.26±0.6	79.28±0.1	**81.40**±0.1	84.62±0.1	88.28±0.1	**89.61**±0.1	**91.49**±0.0
ASPEST with DANN (ours)	**78.14**±0.4	83.33±0.5	93.61±0.0	**79.33**±0.1	81.23±0.1	84.21±0.1	**88.36**±0.2	89.32±0.1	91.26±0.0
ASPEST with CDAN (ours)	77.75±0.3	83.68±0.5	**94.44**±0.3	79.27±0.0	81.30±0.2	**84.76**±0.1	88.35±0.1	89.59±0.0	91.41±0.0

Table E.21: Results of evaluating DE with UDA and ASPEST with UDA on Otto. The mean and std of each metric over three random runs are reported (mean±std). All numbers are percentages. **Bold** numbers are superior results.

F APPENDIX FOR CHAPTER 8

F.1 Proofs for Section 8.3.1

Theorem F.1 (Restatement of Theorem 8.1)**.** *If $\ell(t) = -t$, then the contrastive loss is equivalent to the PCA objective on ϕ_{z_R}:*

$$\mathbb{E}\left[\ell\left(\phi(x)^\top[\phi(x^+) - \phi(x^-)]\right)\right] = -\mathbb{E}\left[\|\phi_{z_R} - \phi_0\|^2\right]. \tag{F.1}$$

If additionally $\phi(x)$ is linear in x, then the contrastive loss is equivalent to the linear PCA objective on data from the distribution $p_{\bar{x}}$ of $\bar{x} = \mathbb{E}_{z_U}[x]$:

$$\mathbb{E}\left[\ell\left(\phi(x)^\top[\phi(x^+) - \phi(x^-)]\right)\right] = -\mathbb{E}\left[\|\phi(\bar{x}) - \phi_0\|^2\right]. \tag{F.2}$$

Proof. We first present some preliminaries for the proof. Recall that in our hidden representation data model $x = g(z)$. The learned representation is $\phi(x) = \phi(g(z)) = \phi \circ g(z)$. For brevity, let us define $\phi(x) = \phi \circ g(z) := h(z)$. Also, the hidden representations corresponding to (x, x^+, x^-) are given by (z, z^+, z^-), where

$$z = [z_R\,;\, z_U],\;\; z^+ = [z_R\,;\, z_U^+],\;\; z^- = [z_R^-\,;\, z_U^-],$$

where z_R and z_R^- are sampled independently from the distribution $\mathcal{D}_R$; and z_U, z_U^+, and z_U^- are sampled independently from the distribution $\mathcal{D}_U$. The expectation of an arbitrary function $f(z, z^+, z^-)$ can be simplified as follows:

$$\begin{aligned}\mathbb{E}_{(z,z^+,z^-)}\left[f(z, z^+, z^-)\right] &= \mathbb{E}_{(z_R, z_R^-, z_U, z_U^+, z_U^-)}\left[f(z, z^+, z^-)\right] \\ &= \mathbb{E}_{(z_R, z_R^-)}\left[\mathbb{E}_{(z_U, z_U^+, z_U^-)}\left[f(z, z^+, z^-) \mid z_R, z_R^-\right]\right].\end{aligned}$$

The second step follows the law of iterated expectations.

The negative expected contrastive loss is

$$
\begin{aligned}
&- \mathbb{E}_{(x,x^+,x^-)} \left[\ell \left(\phi(x)^\top [\phi(x^+) - \phi(x^-)] \right) \right] && \text{(F.3)} \\
&= -\mathbb{E}_{(z,z^+,z^-)} \left[\ell \left(\phi(g(z))^\top [\phi(g(z^+)) - \phi(g(z^-))] \right) \right] && \text{(F.4)} \\
&= \mathbb{E}_{(z,z^+,z^-)} \left[h(z)^\top [h(z^+) - h(z^-)] \right] && \text{(F.5)} \\
&= \mathbb{E}_{(z_R,z_R^-)} \left[\mathbb{E} \left[h(z)^\top [h(z^+) - h(z^-)] \mid z_R, z_R^- \right] \right] && \text{(F.6)} \\
&= \mathbb{E}_{(z_R,z_R^-)} \left[\mathbb{E} \left[h(z) \mid z_R \right]^\top \left(\mathbb{E} \left[h(z^+) \mid z_R \right] - \mathbb{E} \left[h(z^-) \mid z_R^- \right] \right) \right] && \text{(F.7)} \\
&= \mathbb{E}_{(z_R,z_R^-)} \left[\mathbb{E} \left[\phi(x) \mid z_R \right]^\top \left(\mathbb{E} \left[\phi(x^+) \mid z_R \right] - \mathbb{E} \left[\phi(x^-) \mid z_R^- \right] \right) \right] && \text{(F.8)} \\
&= \mathbb{E}_{(z_R,z_R^-)} \left[\phi_{z_R}^\top \left(\phi_{z_R} - \phi_{z_R^-} \right) \right]. && \text{(F.9)}
\end{aligned}
$$

The second equality follows from the choice of loss $\ell(t) = -t$, and the fourth equality follows from the fact that z_U, z_U^+, and z_U^- are sampled independently from the distribution $\mathcal{D}_U$. Also, we have defined $\phi_{z_R} := \mathbb{E}[\phi(x) \mid z_R]$.

Denote the centered representation as $\bar{\phi}_{z_R} = \phi_{z_R} - \phi_0$. Then we have

$$
\begin{aligned}
&- \mathbb{E}_{(x,x^+,x^-)} \left[\ell \left(\phi(x)^\top [\phi(x^+) - \phi(x^-)] \right) \right] && \text{(F.10)} \\
&= \mathbb{E}_{(z_R,z_R^-)} \left[\phi_{z_R}^\top \left(\phi_{z_R} - \phi_{z_R^-} \right) \right] && \text{(F.11)} \\
&= \mathbb{E}_{(z_R,z_R^-)} \left[(\bar{\phi}_{z_R} + \phi_0)^\top \left(\bar{\phi}_{z_R} + \phi_0 - \bar{\phi}_{z_R^-} - \phi_0 \right) \right] && \text{(F.12)} \\
&= \mathbb{E}_{(z_R,z_R^-)} \left[(\bar{\phi}_{z_R} + \phi_0)^\top \left(\bar{\phi}_{z_R} - \bar{\phi}_{z_R^-} \right) \right] && \text{(F.13)} \\
&= \mathbb{E}_{(z_R,z_R^-)} \left[\bar{\phi}_{z_R}^\top \bar{\phi}_{z_R} - \bar{\phi}_{z_R}^\top \bar{\phi}_{z_R^-} \right] + \mathbb{E}_{(z_R,z_R^-)} \left[\phi_0^\top \left(\bar{\phi}_{z_R} - \bar{\phi}_{z_R^-} \right) \right]. && \text{(F.14)}
\end{aligned}
$$

Since $\bar{\phi}_{z_R}$ and $\bar{\phi}_{z_R^-}$ are independent with mean 0, we have $\mathbb{E}_{(z_R,z_R^-)}[\bar{\phi}_{z_R}^\top \bar{\phi}_{z_R^-}] = 0$, $\mathbb{E}_{(z_R,z_R^-)}[\phi_0^\top \bar{\phi}_{z_R}] = 0$, and $\mathbb{E}_{(z_R,z_R^-)}[\phi_0^\top \bar{\phi}_{z_R^-}] = 0$. Therefore,

$$
\begin{aligned}
&- \mathbb{E}_{(x,x^+,x^-)} \left[\ell \left(\phi(x)^\top [\phi(x^+) - \phi(x^-)] \right) \right] && \text{(F.15)} \\
&= \mathbb{E}_{z_R} \left[\bar{\phi}_{z_R}^\top \bar{\phi}_{z_R} \right] && \text{(F.16)} \\
&= \mathbb{E}_{z_R} \left[\| \bar{\phi}_{z_R} \|^2 \right] && \text{(F.17)} \\
&= \mathbb{E}_{z_R} \left[\| \phi_{z_R} - \phi_0 \|^2 \right], && \text{(F.18)}
\end{aligned}
$$

which is the PCA objective on the mean representation ϕ_{z_R}.

If additionally $\phi(x)$ is linear in x, then

$$
\begin{aligned}
&- \mathbb{E}_{(x,x^+,x^-)} \left[\ell \left(\phi(x)^\top [\phi(x^+) - \phi(x^-)] \right) \right] && \text{(F.19)} \\
&= \mathbb{E}_{z_R} \left[\|\phi_{z_R} - \phi_0\|^2 \right] && \text{(F.20)} \\
&= \mathbb{E}_{\bar{x}} \left[\|\phi(\bar{x}) - \phi(x_0)\|^2 \right] && \text{(F.21)}
\end{aligned}
$$

which is the linear PCA objective on the data from the distribution of $\bar{x} = \mathbb{E}[x|z_R]$. □

Theorem F.2 (Restatement of Theorem 8.2)**.** *Under Assumptions* (***A1***)(***A2***)(***A3***)*:*

(1) *The optimal representation ϕ^* does not encode z_U: $\phi^* \circ g(z)$ is independent of z_U.*

(2) *For any invariant feature $i \in R$, there exists $B_i > 0$ such that as long as the representations' norm $B_r \geqslant B_i$, the optimal representation encodes z_i. Furthermore, if z_R is discrete, then B_i is monotonically decreasing in $\Pr[z_{R\setminus\{i\}} = z^-_{R\setminus\{i\}}, z_i \neq z^-_i]$, the probability that in z_R and z^-_R, the i-th feature varies while the others remain the same.*

Proof. (1) Recall that

$$
\phi_{z_R} = \mathbb{E}[\phi \circ g(z) \mid z_R], \quad \phi_0 = \mathbb{E}_z[\phi \circ g(z)] = \mathbb{E}_{z_R}[\phi_{z_R}]. \tag{F.22}
$$

Then the contrastive loss at pre-training is:

$$
\begin{aligned}
&\mathbb{E}_{(x,x^+,x^-)} \left[\ell \left(\phi(x)^\top [\phi(x^+) - \phi(x^-)] \right) \right] && \text{(F.23)} \\
&= \mathbb{E}_{(z,z^+,z^-)} \left[\ell \left((\phi \circ g(z))^\top (\phi \circ g(z^+) - \phi \circ g(z^-)) \right) \right] && \text{(F.24)} \\
&= \mathbb{E}_{(z_R,z^-_R)} \left[\mathbb{E} \left[\ell \left((\phi \circ g(z))^\top (\phi \circ g(z^+) - \phi \circ g(z^-)) \right) \mid z_R, z^-_R \right] \right] && \text{(F.25)} \\
&\geqslant \mathbb{E}_{(z_R,z^-_R)} \left[\ell \left(\mathbb{E} \left[(\phi \circ g(z))^\top (\phi \circ g(z^+) - \phi \circ g(z^-)) \mid z_R, z^-_R \right] \right) \right] && \text{(F.26)} \\
&= \mathbb{E}_{(z_R,z^-_R)} \left[\ell \left(\mathbb{E}[\phi \circ g(z) \mid z_R]^\top \left(\mathbb{E}[\phi \circ g(z^+) \mid z_R] - \mathbb{E}[\phi \circ g(z^-) \mid z^-_R] \right) \right) \right] && \text{(F.27)} \\
&= \mathbb{E}_{(z_R,z^-_R)} \left[\ell \left(\phi^\top_{z_R} \phi_{z_R} - \phi^\top_{z_R} \phi_{z^-_R} \right) \right], && \text{(F.28)}
\end{aligned}
$$

where the inequality comes from the convexity of $\ell(z)$ and Jensen's inequality applied to the inner expectation. The inequality becomes equality when the representation function ϕ is invariant to the spurious features z_U, i.e., with probability 1 over the distribution, $\phi \circ g(z) = \phi_{z_R}$. Therefore, the spurious features z_U are not encoded in the optimal representation, proving the first part.

(2) First consider the case when z has discrete values from a finite set. When the generative function $g(z)$ is not independent of z_i, we assume for contradiction that the optimal representation ϕ is independent of z_i. From (1), we know that it is independent of z_U. So there exists an f such that $\phi \circ g(z) = f(z_{R\setminus\{i\}})$. Without loss of generality, suppose $U = \emptyset$, then $\phi \circ g(z) = f(z_{-i})$.

Since the generative function $g(z)$ is not independent of z_i, there exist z and z^-, such that $z_{-i} = z^-_{-i}$, $z_i \neq z^-_i$, $g(z) \neq g(z^-)$, and z, z^- have non-zero probabilities. So $\Pr[z_{-i} = z^-_{-i}, z_i \neq z^-_i] > 0$.

Now construct a new representation function $\bar{\phi} \in \mathbb{R}^{k+n}, n = |\mathcal{Z}|$ such that $\bar{\phi} \circ g(z) = h(z)$ as follows :

$$h(z) = \left[\sqrt{1-\alpha^2} f(z_{-i}), \quad \alpha \|f(z_{-i})\| \mathbf{I}_z\right] \tag{F.29}$$

where $\mathbf{I}_z$ is the one-hot encoding of the value z. Note that $\bar{\phi}$ still satisfies that norm bound since $\|\bar{\phi}(x)\| = \|h(z)\| = \|f(z_{-i})\|$. We next show that the contrastive loss of $\bar{\phi}$ can be smaller than that of ϕ, leading to a contradiction and finishing the proof.

The contrastive loss of $\bar{\phi}$ (using the fact that $z^+ = z$ when $U = \emptyset$) is

$$\mathbb{E}_{(z,z^-)}\left[\ell\left(h(z)^\top h(z) - h(z)^\top h(z^-)\right)\right] \tag{F.30}$$

$$= \mathbb{E}_{(z,z^-)}\left[\ell\left(h(z)^\top h(z) - h(z)^\top h(z^-)\right) \mid z \neq z^-\right] \Pr[z \neq z^-] \tag{F.31}$$

$$+ \ \mathbb{E}_{z,z^-}[\ell(0)] \Pr[z = z^-].$$

We only need to consider the first term.

$$\mathbb{E}_{(z,z^-)}\left[\ell\left(h(z)^\top h(z) - h(z)^\top h(z^-)\right) \mid z \neq z^-\right] \Pr[z \neq z^-] \tag{F.32}$$

$$= \mathbb{E}_{(z,z^-)}\Big[\underbrace{\ell\left(\|f(z_{-i})\|^2 - (1-\alpha^2) f(z_{-i})^\top f(z_{-i}^-)\right)}_{T_1} \mid z_{-i} \neq z_{-i}^-\Big] \Pr[z_{-i} \neq z_{-i}^-]$$

$$+ \mathbb{E}_{(z,z^-)}\Big[\underbrace{\ell\left(\alpha^2\|f(z_{-i})\|^2\right)}_{T_2} \mid z_{-i} = z_{-i}^-,\, z_i \neq z_i^-\Big] \Pr[z_{-i} = z_{-i}^-,\, z_i \neq z_i^-]. \tag{F.33}$$

When $\alpha = 0$, the above reduces to the corresponding terms for ϕ, so we would like to show that there exists non-zero α that leads to smaller loss values.

Recall that $\ell(\cdot)$ is decreasing by property (A3). Let $\alpha = \sqrt{1/2}/B_r$, where $B_r = \|f(z_{-i})\|$. Then when switching from ϕ to $\bar{\phi}$, T_2 goes from $\ell(0)$ to $\ell(1/2)$, a constant reduction. For T_1, if $f(z_{-i})^\top f(z_{-i}^-)$ is positive, then T_1 decreases; if $f(z_{-i})^\top f(z_{-i}^-)$ is negative, then T_1 increases from $\ell(B_r^2 - f(z_{-i})^\top f(z_{-i}^-))$ to $\ell(B_r^2 - f(z_{-i})^\top f(z_{-i}^-) + \alpha^2 f(z_{-i})^\top f(z_{-i}^-))$. Note that $|\alpha^2 f(z_{-i})^\top f(z_{-i}^-)| \leqslant 1$ (by the Cauchy-Schwarz inequality); so the increase in T_1 diminishes when B_r grows, by the property (A3) of ℓ. Then when B_r is large enough, the increase in T_1 is smaller than the decrease in T_2. So from ϕ to $\bar{\phi}$, the contrastive loss decreases, contradicting that ϕ is optimal. Finally, since the reduction in (F.34) is smaller when $\Pr[z_{-i} = z_{-i}^-, z_i \neq z_i^-]$ is smaller, then B_i needs to be larger. So B_i is monotonically decreasing in $\Pr[z_{-i} = z_{-i}^-, z_i \neq z_i^-]$.

Now consider the general case when z may not be from a finite set. For any $\epsilon_0 > 0$, there exists a ℓ_2 ball $\mathcal{B}$ of bounded radius such that the probability of z outside the ball is at most ϵ_0. Since $\phi \circ g$'s are regular by assumption, there exists a partition $\mathcal{Z} \cap \mathcal{B}$ into finitely many subsets such that in each subset and for each $\phi \circ g$, the function value varies by at most ϵ_0. Construct a new distribution $\mathcal{D}_z'$ for z: select a representative point in each subset, and put a probability mass to it equal to that of the original distribution $\mathcal{D}_z$ in this subset, and normalize the probabilities over the subsets. The new distribution is over a finite set so the above argument holds. Furthermore, the difference in the T_1 term for $\mathcal{D}_z'$ and $\mathcal{D}_z$ can be made arbitrarily small by choosing sufficiently small ϵ_0; similarly for T_2. Then the

argument also holds for $\mathcal{D}_z$, which completes the proof for the general case. □

F.1.1 Inductive Biases are Needed for Analyzing Prediction Success

We have analyzed what features are encoded in the representation. However, encoding the information does not equate to good prediction performance, in particular, with linear predictors. Recently, Saunshi et al. demonstrated that existing analyses that ignore the inductive biases of the model and algorithm cannot adequately explain the prediction success, and provided examples where such analysis can lead to vacuous bounds. One may wonder if our hidden representation data model can provide inductive biases that avoid such vacuous bounds. Unfortunately, similar issues as in Saunshi et al. (2022) remain.

To illustrate that inductive biases are still needed in our data model, consider the following simple example. Suppose $z_R \in \{-1, 1\}^2$ and can be recovered from x; the label y is simply the first coordinate in z_R. Suppose the representation satisfies $\phi(x) \in \mathbb{R}^2$, $\|\phi(x)\| = 1$, and contrastive learning uses the logistic loss $\ell(z)$. Let $\phi(x)$ be such that $\phi \circ g(z) = h(z_R)$, and $h((-1,-1)) = (-1,0), h((-1,1)) = (1,0), h((1,-1)) = (0,-1), h((1,1)) = (0,1)$. It can be verified that this ϕ is optimal for the contrastive loss. However, on the representation ϕ, the classification is an XOR-problem (Fig. F.1), for which there is no non-trivial error bound for linear predictors. This contradicts the success of linear probing in practice.

Furthermore, some restrictions on the data distributions are also needed. Suppose *all* optimal representations are linearly separable with certain inductive biases on the representation function class. Suppose the label y depends on z_R. Without restrictions on the labeling function, one can consider a random $y \in \{-1, +1\}$ over any z_R. Then for any linear predictor on any optimal representation, in expectation the error is $1/2$, so there is always a labeling function for which no non-trivial error can be achieved. Our analysis thus requires restrictions on the dependence of the label on z_R (in particular, we will assume linear dependence).

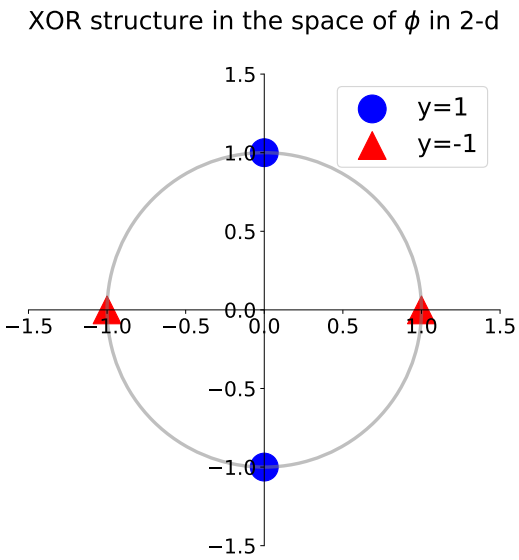


Figure F.1: A two-dim example of XOR structure in the space of ϕ.

F.2 Proofs and More Analysis for Section 8.3.2

F.2.1 Lemmas for a more general setting

We will prove the results in a more general setting, where the mixture can be uneven and the variances of different types of features can be different. The results in Section 8.3.2 then follow from these lemmas.

In the more general setting, the diverse pre-training data is a mixture of data from T different tasks $\mathcal{D}_t$'s, while the target task is one of the tasks. In the mixture, the task $\mathcal{D}_t$ has weight $w_t > 0$ and $\sum_{t=1}^{T} w_t = 1$. All tasks share a public feature set S of size s, and each task $\mathcal{D}_t$ additionally owns a private disjoint feature set P_t of size $r - s$, i.e., $P_t \cap S = \emptyset$ for $t \in [T]$ and $P_{t_1} \cap P_{t_2} = \emptyset$ for $t_1 \neq t_2$. The invariant features for $\mathcal{D}_t$ are then $R_t = S \cup P_t$. All invariant features are $\cup_{t=1}^{T} R_t \subseteq R$, $k := |R|$, and spurious features are $U = [d] \setminus R$. In task $\mathcal{D}_t$, the positive pairs (x, x^+) are

generated as follows:

$$z_S \sim \mathcal{N}(0, \sigma_{S,t}^2 I), z_{P_t} \sim \mathcal{N}(0, \sigma_{R,t}^2 I), z_{R \setminus R_t} = 0, \tag{F.34}$$

$$z_U \sim \mathcal{N}(0, \sigma_{U,t}^2 I), z = [z_R; z_U], \quad x = g(z), \tag{F.35}$$

$$z_U^+ \sim \mathcal{N}(0, \sigma_{U,t}^2 I), z^+ = [z_R; z_U^+], \quad x^+ = g(z^+), \tag{F.36}$$

and x^- is simply an i.i.d. copy from the same distribution as x. In practice, multiple independent negative examples are used, and thus we consider the following contrastive loss

$$\min_{\phi \in \Phi} \mathbb{E}_{(x,x^+)} \left[\ell \left(\phi(x)^\top (\phi(x^+) - \mathbb{E}_{x^-} \phi(x^-)) \right) \right] \tag{F.37}$$

to pre-train a representation ϕ. Then, when using ϕ for prediction in the target task $\mathcal{D}_t$, the predictor class should contain a predictor matching the ground-truth label, so consider the class:

$$\mathcal{F}_{\phi,t} = \{ f(z) = u_t^\top z : u_t \in \mathbb{R}^k, \|u_t\| \leqslant B_{\phi,t} \} \tag{F.38}$$

where $B_{\phi,t}$ is the minimum value such that there exists $u_t \in \mathcal{F}_{\phi,t}$ with $y = u_t^\top \phi(x)$ on $\mathcal{D}_t$.

Recall that we assume a linear data model and linear representation functions ϕ:

- x is linear in z: $x = g(z) = Mz$ where $M \in \mathbb{R}^{d \times d}$ is an orthonormal dictionary. The label in task $\mathcal{D}_t$ is linear in its invariant features $y = (u_t^*)^\top z_{R_t}$ for some $u_t^* \in \mathbb{R}^r$.

- The representations are linear functions with weights of bounded spectral and Frobenius norms:

$$\Phi = \{ \phi(x) = Wx : W \in \mathbb{R}^{k \times d}, \|W\| \leqslant 1, \|W\|_F \leqslant \sqrt{r} \}.$$

  Here the norm bounds are chosen to be the minimum values to allow recov-

ering the invariant features in the target task, i.e., there exists $\phi \in \Phi$ such that $\phi(x) = [z_{R_t}; \mathbf{0}]$.

Lemma F.3. *Consider the above setting. Let* $\alpha, \alpha_t (t \in [T])$ *be the optimizer for*

$$\min_{\tilde{\alpha}, \tilde{\alpha}_1, \ldots, \tilde{\alpha}_T} \sum_{t=1}^{T} w_t \mathbb{E}\left[\ell\left(\tilde{\alpha}\sigma_{S,t}^2 Z + \tilde{\alpha}_t \sigma_{R,t}^2 Z_t\right)\right], \tag{F.39}$$

$$\text{subject to} \quad \tilde{\alpha} s + \sum_{t=1}^{T} \tilde{\alpha}_t (r - s) \leqslant r, \tag{F.40}$$

$$\tilde{\alpha}, \tilde{\alpha}_t \in [0, 1], \tag{F.41}$$

where $Z \sim \chi_s^2$ *and* $Z_t \sim \chi_{r-s}^2$.

Then the optimal representation $\phi^*(x)$ *the loss (F.37) in contrastive learning satisfies* $\phi^*(x) = W^* x$ *with any* W^* *of the form:*

$$W^* = [QA^*, \boldsymbol{0}]M^{-1} \tag{F.42}$$

where $Q \in \mathbb{R}^{k \times k}$ *is any orthonormal matrices,* A^* *is a* $k \times k$ *diagonal matrix with*

$$A_{jj}^* = \begin{cases} \sqrt{\alpha} & \text{if } j \in S, \\ \sqrt{\alpha_t} & \text{if } j \in P_t, \\ 0 & \text{otherwise,} \end{cases} \tag{F.43}$$

and the matrix of zeros has size $k \times (d - k)$.

Proof. For each $\mathcal{D}_t$,

$$\begin{aligned}
&\mathbb{E}_{(x,x^+)}\left[\ell\left(\phi(x)^\top[\phi(x^+)-\mathbb{E}_{x^-}\phi(x^-)]\right)\right] && \text{(F.44)}\\
&=\mathbb{E}_{(z,z^+)}\left[\ell\left((WMz)^\top(WMz^+-\mathbb{E}_{z^-}[WMz^-])\right)\right] && \text{(F.45)}\\
&=\mathbb{E}_{(z,z^+)}\left[\ell\left(z^\top(M^\top W^\top WM)(z^+-\mathbb{E}_{z^-}[z^-])\right)\right] && \text{(F.46)}\\
&\geqslant \mathbb{E}_{z_R}\left[\ell\left((\mathbb{E}_{z_U}[z])^\top M^\top W^\top WM(\mathbb{E}_{z_U^+}[z^+]-\mathbb{E}_{z^-}[z^-])\right)\right] && \text{(F.47)}\\
&=\mathbb{E}_{z_R}\left[\ell\left([z_R;\mathbf{0}]^\top M^\top W^\top WM([z_R;\mathbf{0}]-0)\right)\right] && \text{(F.48)}\\
&=\mathbb{E}_{z_R}\left[\ell\left(\|WM[z_R;\mathbf{0}]\|^2\right)\right] && \text{(F.49)}
\end{aligned}$$

where the inequality comes from the convexity of $\ell(t)$ and Jensen's inequality. Similar to Theorem 8.2, the equality holds if and only if WMz does not depend on z_U and WMz^+ does not depend on z_U^+, so the optimal solution should satisfy this condition.

Let $WM = [A_R, A_U]$ where $A_R \in \mathbb{R}^{k\times k}, A_U \in \mathbb{R}^{k\times(d-k)}$. By rotational invariance of z_S, and z_{P_t}, without loss of generality, we can assume $A_R = QA$ where A is a diagonal matrix with diagonal entries a_{jj}'s and Q is any orthonormal matrix. Furthermore, $A_U = 0$ in the optimal solution since it does not affect the loss but only decreases the norm bound on A_R. So on data from the task $\mathcal{D}_t$,

$$\mathbb{E}_{\mathcal{D}_t}\left[\ell\left(\|WM[z_R;\mathbf{0}]\|^2\right)\right] = \mathbb{E}_{z_{R_t}}\left[\ell\left(\sum_{j\in R_t} a_{jj}^2 z_j^2\right)\right]. \tag{F.50}$$

Then on the mixture,

$$\mathbb{E}_{(x,x^+)}\left[\ell\left(\phi(x)^\top[\phi(x^+)-\mathbb{E}_{x^-}\phi(x^-)]\right)\right] \tag{F.51}$$

$$\geqslant \sum_{t=1}^{T} w_t \mathbb{E}_{\{z_j\}}\left[\ell\left(\sum_{j\in R_t} a_{jj}^2 z_j^2\right)\right] \tag{F.52}$$

$$= \sum_{t=1}^{T} w_t \mathbb{E}_{\{\tilde{z}_j\sim\mathcal{N}(0,1)\}}\left[\ell\left(\sum_{j\in S} a_{jj}^2\sigma_{S,t}^2\tilde{z}_j^2 + \sum_{j\in P_t} a_{jj}^2\sigma_{R,t}^2\tilde{z}_j^2\right)\right] \tag{F.53}$$

$$:= g(\{a_{jj}\}), \tag{F.54}$$

where each $\tilde{z}_j$ is a random variable drawn from standard Gaussian.

Now consider the minimum of the function $g(\{a_{jj}\})$ on the right hand side, under the constraints that $|a_{jj}| \leqslant 1$ and $\sum_j a_{jj}^2 \leqslant r$. Before finishing the proof of Lemma F.3, we have the following claim for this optimization.

Claim F.4. *There exist α, α_t satisfying $0 \leqslant \alpha, \alpha_t \leqslant 1$ and $\alpha s + \sum_{t=1}^{T}\alpha_t(r-s) = \sum_j a_{jj}^2 \leqslant r$, such that the minimum of the above optimization (F.54) is achieved when $a_{jj}^2 = \alpha$ for any $j \in S$, and $a_{jj}^2 = \alpha_t$ for any $j \in P_t$ and $t \in [T]$.*

Proof. We need to prove that to achieve the minimum,

(1) $a_{\ell\ell}^2 = a_{\ell'\ell'}^2$ for any $\ell \neq \ell' \in S$;

(2) $a_{\ell\ell}^2 = a_{\ell'\ell'}^2$ for any $\ell \neq \ell' \in P_t$ and any $t \in [T]$;

For (1): By symmetry of z_j's and the convexity of $\ell(\cdot)$, for any $t \in [T]$,

$$\mathbb{E}\left[\ell\left(\sum_{j\in R_t} a_{jj}^2 z_j^2\right)\right] \tag{F.55}$$

$$= \frac{1}{2}\mathbb{E}\left[\ell\left(\sum_{j\in S, j\neq \ell, j\neq \ell'} a_{jj}^2 z_j^2 + a_{\ell\ell}^2 z_\ell^2 + a_{\ell'\ell'}^2 z_{\ell'}^2 + \sum_{j\in P_t} a_{jj}^2 z_j^2\right)\right] \tag{F.56}$$

$$+ \frac{1}{2}\mathbb{E}\left[\ell\left(\sum_{j\in S, j\neq \ell, j\neq \ell'} a_{jj}^2 z_j^2 + a_{\ell\ell}^2 z_{\ell'}^2 + a_{\ell'\ell'}^2 z_\ell^2 + \sum_{j\in P_t} a_{jj}^2 z_j^2\right)\right] \tag{F.57}$$

$$\geqslant \mathbb{E}\left[\ell\left(\sum_{j\in S, j\neq \ell, j\neq \ell'} a_{jj}^2 z_j^2 + \frac{a_{\ell\ell}^2 + a_{\ell'\ell'}^2}{2} z_{\ell'}^2 + \frac{a_{\ell\ell}^2 + a_{\ell'\ell'}^2}{2} z_\ell^2 + \sum_{j\in P_t} a_{jj}^2 z_j^2\right)\right]. \tag{F.58}$$

Then

$$g(\{a_{jj}\}) \tag{F.59}$$

$$\geqslant \sum_{t=1}^{T} w_t \mathbb{E}\left[\ell\left(\sum_{j\in S, j\neq \ell, j\neq \ell'} a_{jj}^2 z_j^2 + \frac{a_{\ell\ell}^2 + a_{\ell'\ell'}^2}{2} z_{\ell'}^2 + \frac{a_{\ell\ell}^2 + a_{\ell'\ell'}^2}{2} z_\ell^2 + \sum_{j\in P_t} a_{jj}^2 z_j^2\right)\right].$$

Therefore, the minimum is achieved when $a_{\ell\ell}^2 = a_{\ell'\ell'}^2$.

A similar argument as above proves statement (2). □

These statements mean that, for any $t \in [T]$, the minimum is achieved when $a_{jj}^2 = \alpha$ for $j \in S$, and $a_{jj}^2 = \alpha_t$ for $j \in P_t$, for some values $\alpha, \alpha_t \geqslant 0$. Let $Z = \sum_{j\in S} \tilde{z}_j^2$, $Z_t = \sum_{j\in P_t} \tilde{z}_j^2$. Then $Z \sim \chi_s^2$ and $Z_t \sim \chi_{r-s}^2$, and we have:

$$g(\{a_{jj}\}) = \sum_{t=1}^{T} w_t \mathbb{E}\left[\ell\left(\sum_{j\in S} \alpha\sigma_{S,t}^2 \tilde{z}_j^2 + \sum_{j\in P_t} \alpha_t \sigma_{R,t}^2 \tilde{z}_j^2\right)\right] \tag{F.60}$$

$$= \sum_{t=1}^{T} w_t \mathbb{E}\left[\ell\left(\alpha\sigma_{S,t}^2 Z + \alpha_t \sigma_{R,t}^2 Z_t\right)\right]. \tag{F.61}$$

Given the constraint $\alpha s + \sum_{t=1}^{T} \alpha_t (r-s) = \sum_j a_{jj}^2 \leqslant r$, $0 \leqslant \alpha, \alpha_t \leqslant 1$, we complete

the proof of Lemma F.3. □

Given this result we can now analyze the generalization error when predicting on the target task $\mathcal{D}_t$.

Lemma F.5. *Consider any* $t \in [T]$. *Let* $\nu_{t,1} = \sum_{j\in S}(u_t^*)_j^2$ *and* $\nu_{t,2} = \sum_{j\in P_t}(u_t^*)_j^2$. *Suppose in* ϕ^* *(calculated in Lemma F.3),* $\alpha, \alpha_t > 0$. *Suppose the prediction loss* ℓ_c *is* L*-Lipschitz.*

Then the Empirical Risk Minimizer $\hat{u}_t \in \mathcal{F}_{\phi^*,t}$ *on* ϕ^* *using* m *labeled data points from* $\mathcal{D}_t$ *has risk*

$$\mathbb{E}_{(x,y)\sim\mathcal{D}_t}[\ell_c(\hat{u}_t{}^\top\phi^*(x), y)] \leqslant 8\sqrt{\frac{2\ln(4/\delta)}{m}} + 4L\sqrt{\frac{1}{m}\left(\frac{\nu_{t,1}}{\alpha}+\frac{\nu_{t,2}}{\alpha_t}\right)} \cdot \left(\sqrt{s\alpha\sigma_{S,t}^2 + (r-s)\alpha_t\sigma_{R,t}^2} + O\left(\sqrt{\frac{\max\{\alpha\sigma_{S,t}^2, \alpha_t\sigma_{R,t}^2\}^2 r}{s\alpha\sigma_{S,t}^2 + (r-s)\alpha_t\sigma_{R,t}^2}}\right)\right).$$

Proof. For any $t \in [T]$, we only need to bound the Rademacher complexity $\mathcal{R}_m(\mathcal{F}_{\phi^*,t})$ of $\mathcal{F}_{\phi^*,t}$; the statement then follows from standard generalization bounds,

$$\mathbb{E}_{(x,y)\sim\mathcal{D}_t}[\ell_c(\hat{u}_t{}^\top\phi^*(x), y)] \leqslant 4L\mathcal{R}_m(\mathcal{F}_{\phi^*,t}) + 8\sqrt{\frac{2\ln(4/\delta)}{m}}.$$

Given the representation ϕ^* in Lemma F.3, to ensure there exists a predictor in $\mathcal{F}_{\phi^*,t}$ matching the ground-truth label, $f(\phi^*(x)) = u_t^\top\phi^*(x) = y = (u_t^*)^\top z_{R_t}$, predictor u_t should satisfy

$$\begin{aligned}
\mathbb{E}_{\mathcal{D}_t}[(\hat{y}-y)^2] = 0 \Leftrightarrow &\forall z_{R_t}, \quad u_t{}^\top[QA^*, \mathbf{0}]M^{-1}M[z_{R_t}; \mathbf{0}; z_U] = u_t^{*\top} z_{R_t} && \text{(F.62)}\\
\Leftrightarrow &\forall z_{R_t}, \quad u_t{}^\top QA^*[z_{R_t}; \mathbf{0}] = u_t^{*\top} z_{R_t} && \text{(F.63)}\\
\overset{(*)}{\Leftrightarrow} &A^*_{1:r,1:r}(Q^\top)_{1:r,1:k}u_t = u_t^* && \text{(F.64)}\\
\Leftrightarrow &\forall v \in \mathbb{R}^r, \quad u_t = Q_{1:k,1:r}(A^*_{1:r,1:r})^{-1}u_t^* + Q_{1:k,r+1:k}v. && \text{(F.65)}
\end{aligned}$$

The $(*)$ is from non-zero variance for z_{R_t}. $u_t = Q_{1:k,1:r}(A^*_{1:r,1:r})^{-1}u_t^*$ is the least-norm optimal solution, so we have $B_{\phi^*,t} = \|Q_{1:k,1:r}(A^*_{1:r,1:r})^{-1}u_t^*\| = \sqrt{\frac{\nu_{t,1}}{\alpha}+\frac{\nu_{t,2}}{\alpha_t}}$. So the

predictor class should be

$$\mathcal{F}_{\phi^*,t} = \left\{ f(\phi^*) = u_t^\top \phi^* : u_t \in \mathbb{R}^k, \|u_t\| \leqslant B_{\phi^*,t} = \sqrt{\frac{\nu_{t,1}}{\alpha} + \frac{\nu_{t,2}}{\alpha_t}} \right\}. \tag{F.66}$$

The empirical Rademacher complexity and Rademacher complexity of $\mathcal{F}_{\phi^*,t}$ with m samples are

$$\begin{aligned}
\hat{\mathcal{R}}_m(\mathcal{F}_{\phi^*,t}) =& \frac{1}{m}\mathbb{E}_\sigma \left[\sup_{f_{u,\phi} \in \mathcal{F}_{\phi^*,t}} \sum_{i=1}^m \sigma_i f_{u,\phi}(x^{(i)}) \right] && \text{(F.67)} \\
=& \frac{1}{m}\mathbb{E}_\sigma \left[\sup_{\|u\| \leqslant B_{\phi^*,t}} \sum_{i=1}^m \sigma_i u_t{}^\top Q A^*[z_{R_t}^{(i)}; \mathbf{0}] \right] && \text{(F.68)} \\
=& \frac{1}{m}\mathbb{E}_\sigma \left[\sup_{\|u\| \leqslant B_{\phi^*,t}} u_t{}^\top \sum_{i=1}^m \sigma_i Q_{1:k,1:r} A^*_{1:r,1:r} z_{R_t}^{(i)} \right] && \text{(F.69)} \\
=& \frac{B_{\phi^*,t}}{m}\mathbb{E}_\sigma \left[\left\| \sum_{i=1}^m \sigma_i Q_{1:k,1:r} A^*_{1:r,1:r} z_{R_t}^{(i)} \right\| \right], && \text{(F.70)}
\end{aligned}$$

$$\begin{aligned}
\mathcal{R}_m(\mathcal{F}_{\phi^*,t}) =& \mathbb{E}_{z_R, z_U} \left[\hat{\mathcal{R}}_m(\mathcal{F}_{\phi^*,t}) \right] && \text{(F.71)} \\
=& \frac{B_{\phi^*,t}}{m}\mathbb{E}_{z_{R_t}^{(i)}} \left[\mathbb{E}_\sigma \left[\left\| \sum_{i=1}^m \sigma_i Q_{1:k,1:r} A^*_{1:r,1:r} z_{R_t}^{(i)} \right\| \right] \right] && \text{(F.72)} \\
=& \frac{B_{\phi^*,t}}{m}\mathbb{E}_{z_{R_t}^{(i)}} \left[\left\| A^*_{1:r,1:r} \sum_{i=1}^m z_{R_t}^{(i)} \right\| \right]. && \text{(F.73)}
\end{aligned}$$

For any $t \in [T]$, define $X_t := A^*_{1:r,1:r} \sum_{i=1}^m z_{R_t}^{(i)}$. Note that for $j \in S$, $X_{t,j} = \alpha \sum_{i=1}^m z_j^{(i)}$ is a Gaussian of mean zero and variance

$$\mathbb{E}[X_{t,j}^2] = \alpha \mathbb{E}\left[\left(\sum_{i=1}^m z_j^{(i)} \right)^2 \right] = \alpha \mathbb{E}\left[\sum_{i=1}^m \left(z_j^{(i)} \right)^2 \right] = m\alpha\sigma_{S,t}^2. \tag{F.74}$$

Similarly, for $j \in P_t$, $X_{t,j} = \alpha_t \sum_{i=1}^m z_j^{(i)}$ is a Gaussian of mean zero and variance

$\mathbb{E}[X_{t,j}^2] = m\alpha_t\sigma_{R,t}^2$. Since $X_{t,j}$ is sub-gaussian, $X_{t,j}^2 - m\alpha\sigma_{S,t}^2$ for $j \in S$ and $X_{t,j}^2 - m\alpha_t\sigma_{R,t}^2$ for $j \in P_t$ are sub-exponential and more precisely

$$\|X_{t,j}^2 - m\alpha\sigma_{S,t}^2\|_{\psi_1} \leqslant C_1\|X_{t,j}^2\|_{\psi_1} = C_1\|X_{t,j}\|_{\psi_2}^2 \leqslant C_2 m\alpha\sigma_{S,t}^2, \quad j \in S, \tag{F.75}$$

$$\|X_{t,j}^2 - m\alpha_t\sigma_{R,t}^2\|_{\psi_1} \leqslant C_1\|X_{t,j}^2\|_{\psi_1} = C_1\|X_{t,j}\|_{\psi_2}^2 \leqslant C_2 m\alpha_t\sigma_{R,t}^2, \quad j \in P_t, \tag{F.76}$$

where C_1, C_2 are absolute constants and $C_2 > 1$. Let

$$K = \max(C_2 m\alpha\sigma_{S,t}^2, C_2 m\alpha_t\sigma_{R,t}^2) = C_2 m \max\{\alpha\sigma_{S,t}^2, \alpha_t\sigma_{R,t}^2\}$$

and

$$\mu := m(s\alpha\sigma_{S,t}^2 + (r-s)\alpha_t\sigma_{R,t}^2).$$

By Bernstein's inequality, we have for every $\gamma \geqslant 0$ that

$$\mathbb{P}\left\{\left|\frac{1}{r}(\|X_t\|^2 - \mu)\right| \geqslant \gamma\right\} \leqslant 2\exp\left[-c\min\left(\frac{\gamma^2}{K^2}, \frac{\gamma}{K}\right) r\right] \tag{F.77}$$

$$\Rightarrow \mathbb{P}\left\{\left|\frac{\|X_t\|^2}{\mu} - 1\right| \geqslant \frac{r\gamma}{\mu}\right\} \tag{F.78}$$

$$\leqslant 2\exp\left[-\frac{c}{C_2^2}\min\left(\frac{\gamma^2}{m^2\max\{\alpha\sigma_{S,t}^2, \alpha_t\sigma_{R,t}^2\}^2}, \frac{\gamma}{m\max\{\alpha\sigma_{S,t}^2, \alpha_t\sigma_{R,t}^2\}}\right) r\right], \tag{F.79}$$

where c is an absolute constant. For all numbers $z \geqslant 0$, we have $|z-1| \geqslant \delta \Rightarrow$

$|z^2-1| \geqslant \max(\delta, \delta^2)$. Thus, for any $\delta \geqslant 0$, we have

$$\mathbb{P}\left\{\left|\frac{\|X_t\|}{\sqrt{\mu}}-1\right| \geqslant \delta\right\} \tag{F.80}$$

$$\leqslant \mathbb{P}\left\{\left|\frac{\|X_t\|_2^2}{\mu}-1\right| \geqslant \max(\delta, \delta^2)\right\} \tag{F.81}$$

$$\leqslant 2\exp\left[-\frac{c}{C_2^2}\min\left(\left(\frac{\mu\max(\delta,\delta^2)}{m\max\{\alpha\sigma_{S,t}^2, \alpha_t\sigma_{R,t}^2\}r}\right)^2, \frac{\mu\max(\delta,\delta^2)}{m\max\{\alpha\sigma_{S,t}^2, \alpha_t\sigma_{R,t}^2\}r}\right) r\right] \tag{F.82}$$

$$\leqslant 2\exp\left[-\frac{c}{C_2^2}\left(\frac{\mu}{m\max\{\alpha\sigma_{S,t}^2, \alpha_t\sigma_{R,t}^2\}r}\right)^2 \min\left(\left(\max(\delta,\delta^2)\right)^2, \max(\delta,\delta^2)\right) r\right] \tag{F.83}$$

$$= 2\exp\left[-\frac{c}{C_2^2}\frac{\mu^2}{m^2\max\{\alpha\sigma_{S,t}^2, \alpha_t\sigma_{R,t}^2\}^2 r}\delta^2\right], \tag{F.84}$$

where the last inequality is from

$$\mu = m(s\alpha\sigma_{S,t}^2 + (r-s)\alpha_t\sigma_{R,t}^2) \leqslant m\max\{\alpha\sigma_{S,t}^2, \alpha_t\sigma_{R,t}^2\}r.$$

Changing variables to $\theta = \delta\sqrt{\mu}$, we obtain the desired sub-gaussian tail

$$\mathbb{P}\{|\|X_t\| - \sqrt{\mu}| \geqslant \theta\} \leqslant 2\exp\left[-\frac{c}{C_2^2}\frac{\mu}{m^2\max\{\alpha\sigma_{S,t}^2, \alpha_t\sigma_{R,t}^2\}^2 r}\theta^2\right]. \tag{F.85}$$

By generalization of integral identity, we have

$$|\mathbb{E}\left[\|X_t\| - \sqrt{\mu}\right]| = \left|\int_0^\infty \mathbb{P}\{\|X_t\| - \sqrt{\mu} > \theta\}d\theta - \int_{-\infty}^0 \mathbb{P}\{\|X_t\| - \sqrt{\mu} < \theta\}d\theta\right| \tag{F.86}$$

$$\leqslant 2\int_0^\infty \mathbb{P}\{|\|X_t\| - \sqrt{\mu}| > \theta\}d\theta \tag{F.87}$$

$$\leqslant 4\int_0^\infty \exp\left[-\frac{c}{C_2^2}\frac{\mu}{m^2\max\{\alpha\sigma_{S,t}^2, \alpha_t\sigma_{R,t}^2\}^2 r}\theta^2\right] d\theta \tag{F.88}$$

$$\leqslant C_3\frac{m\max\{\alpha\sigma_{S,t}^2, \alpha_t\sigma_{R,t}^2\}\sqrt{r}}{\sqrt{\mu}}, \tag{F.89}$$

where C_3 is an absolute constant. Thus, we have

$$\left|\mathcal{R}_m(\mathcal{F}_{\phi^*,t}) - \sqrt{\frac{1}{m}\left(\frac{\nu_{t,1}}{\alpha}+\frac{\nu_{t,2}}{\alpha_t}\right)\left(s\alpha\sigma_{S,t}^2+(r-s)\alpha_t\sigma_{R,t}^2\right)}\right| \tag{F.90}$$

$$= \frac{B_{\phi^*,t}}{m}\left|\mathbb{E}\left[\|X_t\|\right]-\sqrt{\mu}\right| \tag{F.91}$$

$$\leqslant O\left(\sqrt{\frac{1}{m}\left(\frac{\nu_{t,1}}{\alpha}+\frac{\nu_{t,2}}{\alpha_t}\right)\frac{\max\{\alpha\sigma_{S,t}^2,\alpha_t\sigma_{R,t}^2\}^2 r}{s\alpha\sigma_{S,t}^2+(r-s)\alpha_t\sigma_{R,t}^2}}\right). \tag{F.92}$$

□

F.2.2 Proofs of Proposition 6 and Proposition 7

Given the lemmas for the general case, we are now ready to prove the results in Proposition 6 and Proposition 7.

Proposition 15 (Restatement of Proposition 6)**.** *Suppose* $\sigma_{S,t}=\sigma_{R,t}=\sigma_{U,t}=1$ *for any* $t\in[T]$*. The representation* ϕ^* *obtained on an even mixture of data from all the tasks* $\{\mathcal{D}_t : 1\leqslant t\leqslant T\}$ *satisfies* $\phi^*\circ g(z)=Q\left(\sum_{j\in S}\sqrt{\alpha}z_j e_j+\sum_{j\in R\setminus S}\sqrt{\beta}z_j e_j\right)$ *for some* $\alpha\in[0,1],\ \beta=\min\left(1,\frac{r-\alpha s}{T(r-s)}\right)$*, where* e_j*'s are the basis vectors and* Q *is any orthonormal matrix.*
The Empirical Risk Minimizer $\hat{u}\in\mathcal{F}_{\phi^*,t}$ *on* ϕ^* *using* m *labeled data points from* $\mathcal{D}_t$ *has risk*

$$\begin{aligned}&\mathbb{E}_{(x,y)\sim\mathcal{D}_t}[\ell_c(\hat{u}^\top\phi^*(x),y)]\\ \leqslant& 4L\sqrt{\frac{1}{m}\left(\frac{\nu_{t,1}}{\alpha}+\frac{\nu_{t,2}}{\beta}\right)}\left(\sqrt{s\alpha+(r-s)\beta}+O\left(\sqrt{\frac{r}{s\alpha+(r-s)\beta}}\right)\right)\\ &+8\sqrt{\frac{2\ln(4/\delta)}{m}}.\end{aligned}$$

Proof. This follows from Lemma F.3, and considering the optimal α,α_t for the

following:

$$g(\{\alpha, \alpha_t\}) = \sum_{t=1}^{T} w_t \mathbb{E}\left[\ell\left(\alpha\sigma_{S,t}^2 Z + \alpha_t \sigma_{R,t}^2 Z_t\right)\right] \tag{F.93}$$

$$= \frac{1}{T}\sum_{t=1}^{T} \mathbb{E}\left[\ell\left(\alpha Z + \alpha_t Z_1\right)\right] \tag{F.94}$$

$$\geqslant \mathbb{E}\left[\ell\left(\alpha Z + Z_1 \sum_{t=1}^{T} \frac{1}{T}\alpha_t\right)\right]. \tag{F.95}$$

The second equation is from that Z_t's follow the same distribution by the symmetry of $\tilde{z}_j$'s. The inequality comes from the convexity of $\ell(t)$ and Jensen's inequality. So the minimum is achieved when $\alpha_t := \beta$ for any $t \in [T]$, leading to

$$g(\{\alpha, \alpha_t\}) = \mathbb{E}\left[\ell\left(\alpha Z + \beta Z_1\right)\right] \tag{F.96}$$

subject to the constraints $\alpha s + T\beta(r-s) \leqslant r$, $0 \leqslant \alpha, \beta \leqslant 1$. Then we get $\phi^* \circ g(z) = W^* M z = Q\left(\sum_{j\in S}\sqrt{\alpha} z_j e_j + \sum_{j\in R\setminus S}\sqrt{\beta} z_j e_j\right)$ for some $\alpha \in [0,1]$, $\beta = \min\left(1, \frac{r-\alpha s}{T(r-s)}\right)$, where e_j's are the basis vectors and Q is any orthonormal matrix.

Finally, the generalization bound follows from Lemma F.5, and that

$$O\left(\sqrt{\frac{\max\{\alpha,\beta\}^2 r}{s\alpha + (r-s)\beta}}\right) = O\left(\sqrt{\frac{r}{s\alpha + (r-s)\beta}}\right). \tag{F.97}$$

This completes the proof. □

Proposition 16 (Restatement of Proposition 7)**.** *Suppose* $\sigma_{S,t} = \sigma_{R,t} = \sigma_{U,t} = 1$. *The representation* ϕ_t^* *obtained on data from* $\mathcal{D}_t$ *satisfies* $\phi_t^* \circ g(z) = Q\left(\sum_{j\in R_t} z_j e_j\right)$ *where* e_j*'s are the basis vectors and* Q *is any orthonormal matrix.*

The Empirical Risk Minimizer $\hat{u} \in \mathcal{F}_{\phi_t^*, t}$ *on* ϕ_t^* *using* m *labeled data points from* $\mathcal{D}_t$ *has*

risk

$$\mathbb{E}_{(x,y)\sim\mathcal{D}_t}[\ell_c(\hat{u}^\top \phi_t^*(x), y)] \leqslant 4L\sqrt{\frac{r}{m}}\|u_t^*\| + 8\sqrt{\frac{2\ln(4/\delta)}{m}}.$$

While on task $\mathcal{D}_i (i \neq t)$, any linear predictor on ϕ_t^* has error at least

$$\min_u \mathbb{E}_{\mathcal{D}_i}[\ell_c(u^\top z_S, y)].$$

Proof. Following Lemma F.3 (with $r = s$), we get $\phi_t^* \circ g(z) = Q\left(\sum_{j\in R_t} z_j e_j\right)$, where e_j's are the basis vectors and Q is any orthonormal matrix. Following the same argument as in the proof of Lemma F.5, we get

$$\mathcal{R}_m(\mathcal{F}_{\phi^*,t}) = \frac{\|u_t^*\|}{m}\mathbb{E}_{z_{R_t}^{(i)}}\left[\left\|\sum_{i=1}^m z_{R_t}^{(i)}\right\|\right] \leqslant \sqrt{\frac{r}{m}}\|u_t^*\|, \tag{F.98}$$

where the last inequality comes from the property of chi-squared distribution expectation. □

F.2.3 Implication for the trade-off

The propositions then imply the trade-off between universality and label efficiency. Below we formalize the example discussed in Section 8.3.2.

Proposition 17 (A specific version of Proposition 6)**.** *Suppose $\sigma_{S,t} = \sigma_{R,t} = \sigma_{U,t} = 1$ for any $t \in [T]$ and $r = 2s$. The representation ϕ^* obtained on an even mixture of data from all the tasks $\{\mathcal{D}_t : 1 \leqslant t \leqslant T\}$ satisfies $\phi^* \circ g(z) = Q\left(\sum_{j\in S} z_j e_j + \sum_{j\in R\setminus S}\sqrt{\frac{1}{T}} z_j e_j\right)$, where e_j's are the basis vectors and Q is any orthonormal matrix.*
The Empirical Risk Minimizer $\hat{u} \in \mathcal{F}_{\phi^*,t}$ on ϕ^* using m labeled data points from $\mathcal{D}_t$ has

risk

$$\mathbb{E}_{(x,y)\sim\mathcal{D}_t}[\ell_c(\hat{u}^\top \phi^*(x), y)] \leqslant$$
$$4L\sqrt{\frac{1}{m}(\nu_{t,1} + T\nu_{t,2})}\left(\sqrt{r\left(\frac{T+1}{2T}\right)} + O(1)\right) + 8\sqrt{\frac{2\ln(4/\delta)}{m}}.$$

Proof. This follows from Proposition 6, and noting that when $r = 2s$, $\alpha = 1$ and $\beta = 1/T$ are the optimal for:

$$g(\{\alpha, \beta\}) = \mathbb{E}\left[\ell\left(\alpha Z + \beta Z_1\right)\right] \tag{F.99}$$
$$= \mathbb{E}\left[\ell\left(\alpha Z + \frac{r - \alpha s}{T(r-s)} Z_1\right)\right] \tag{F.100}$$
$$= \mathbb{E}\left[\ell\left(\alpha Z + \frac{2-\alpha}{T} Z_1\right)\right] \tag{F.101}$$

subject to the constraints $\alpha s + T\beta(r - s) \leqslant r$, $0 \leqslant \alpha, \beta \leqslant 1$. To see this, note that $Z \sim \chi^2_s$ and $Z_1 \sim \chi^2_{r-s} = \chi^2_s$ follow the same distribution, so $\alpha Z + \frac{2-\alpha}{T} Z_1$ for $\alpha = 1$ will stochastically dominate its value for other $\alpha \in [0, 1)$. The optimal is then achieved when $\alpha = 1$ and $\beta = \frac{2-\alpha}{T} = \frac{1}{T}$. □

F.2.4 Improving the Trade-off by Contrastive Regularization

The above analysis shows that contrastive learning a representation on unlabeled data from the target task can help in prediction on this target task. This suggests that given a representation ϕ^* pre-trained on diverse data, one can fine-tune it by contrastive learning on some unlabeled data from the target task to get a representation that can lead to better prediction on the target task. In the following, we will formally show that this is indeed the case for the illustrative example in Section 8.3.2.

Recall that in this example, $\sigma_{S,t} = \sigma_{R,t} = \sigma_{U,t} = 1$, $r = 2s$, and $\nu_{t,1} = \nu_{t,2}$. The representation ϕ^* obtained on an even mixture of data from all the tasks $\{\mathcal{D}_t : 1 \leqslant t \leqslant T\}$ satisfies $\phi^* \circ g(z) = Q\left(\sum_{j\in S} \sqrt{\alpha} z_j e_j + \sum_{j\in R\setminus S} \sqrt{\beta} z_j e_j\right)$, where

e_j's are the basis vectors and Q is any orthonormal matrix, and $\alpha = 1, \beta = \frac{1}{T}$.

Now, suppose we are given unlabeled data from $\mathcal{D}_t$, and we use them to fine-tune $\phi^*(x) = W^* x$ by contrastive learning on these unlabeled data. That is, we find W near W^* to minimize the contrastive loss on the unlabeled data from $\mathcal{D}_t$:

$$\min_{\phi(x)=Wx} \mathbb{E}\left[\ell\left(\phi(x)^\top(\phi(x^+) - \mathbb{E}_{x^-}\phi(x^-))\right)\right] \tag{F.102}$$

$$\text{subject to} \quad \|W - W^*\|_F \leqslant \gamma, \|W\| \leqslant 1. \tag{F.103}$$

for some small $\gamma > 0$.

Proposition 18. *For (F.102), $\phi^*_{CR,t}$ satisfying the following on x from $\mathcal{D}_t$ is an optimal representation:*

$$\phi^*_{CR,t} \circ g(z) = Q\left(\sum_{j\in S}\sqrt{\alpha} z_j e_j + \sum_{j\in P_t}\sqrt{\beta} z_j e_j\right)$$

where $\sqrt{\alpha} = 1, \sqrt{\beta} = \min\left(1, \sqrt{\frac{1}{T}} + \frac{\gamma}{\sqrt{s}}\right)$.

Proof. Following the argument in Lemma F.3, we still have that $\phi^*_{CR,t}(x) = Wx$ where $W = Q_2[A_2; \mathbf{0}]M^{-1}$ for any orthonormal matrix Q_2 and some diagonal matrix $A_2 = \text{diagonal}(a_{jj})$, with $a_{jj} = \sqrt{\alpha}$ for $j \in S$ and $a_{jj} = \sqrt{\beta}$ for $j \in P_t$ for some $\alpha, \beta \in [0, 1]$. And the contrastive loss is:

$$\mathbb{E}\left[\ell\left(\phi(x)^\top(\phi(x^+) - \mathbb{E}_{x^-}\phi(x^-))\right)\right] = \mathbb{E}\left[\ell\left(\sum_{j\in R_t} a_{jj}^2 z_j^2\right)\right] \tag{F.104}$$

$$= \mathbb{E}\left[\ell\left(\alpha\sum_{j\in S} z_j^2 + \beta\sum_{j\in P_t} z_j^2\right)\right]. \tag{F.105}$$

Recall that $\phi^*(x) = W^* x$ with $W = Q[A; \mathbf{0}]M^{-1}$ for any orthonormal matrix Q and some diagonal matrix A, with $A_{jj} = 1$ for $j \in S$ and $A_{jj} = \sqrt{1/T}$ for $j \in R_i \setminus S$

for any $i \in [T]$. Then

$$\|W - W^*\|_F = \|Q_2[A_2; \mathbf{0}]M^{-1} - Q[A; \mathbf{0}]M^{-1}\|_F \tag{F.106}$$
$$= \|Q_2 A_2 - QA\|_F \tag{F.107}$$
$$= \|A_2 - Q_2^{-1}QA\|_F. \tag{F.108}$$

Since $Q_2^{-1}Q$ is a rotation and A, A_2 are diagonal, we can always set $Q_2 = Q$ without increasing $\|W - W^*\|_F$. Then

$$\|W - W^*\|_F^2 = \|A_2 - A\|_F^2 \tag{F.109}$$
$$= s(\sqrt{\alpha} - 1)^2 + s(\sqrt{\beta} - \sqrt{1/T})^2 + \sum_{j \in P_i, i \neq t} ((A_2)_{jj} - \sqrt{1/T})^2. \tag{F.110}$$

To minimize the contrastive loss, we need α, β to be as large as possible, subject to $\|W - W^*\|_F^2 \leqslant \gamma^2$, and $\alpha, \beta, (A_2)_{jj}^2 \in [0, 1]$. The optimal is then achieved when $\alpha = 1, \sqrt{\beta} = \min\left(1, \sqrt{\frac{1}{T}} + \frac{\gamma}{\sqrt{s}}\right)$, and $(A_2)_{jj} = \sqrt{1/T}$ for $j \in P_i, i \neq t$. □

Now, recall that by Proposition 6, the ERM has risk:

$$\begin{aligned}
&\mathbb{E}_{(x,y)\sim\mathcal{D}_t}[\ell_c(\hat{u}^\top \phi^*(x), y)] \\
\leqslant& 4L\sqrt{\frac{1}{m}\left(\frac{\nu_{t,1}}{\alpha} + \frac{\nu_{t,2}}{\beta}\right)}\left(\sqrt{s\alpha + (r-s)\beta} + O\left(\sqrt{\frac{r}{s\alpha + (r-s)\beta}}\right)\right) \\
&+ 8\sqrt{\frac{2\ln(4/\delta)}{m}} \\
=& 4L\sqrt{\frac{1}{m}\left(\frac{\|u_t^2\|^2}{2\alpha} + \frac{\|u_t^2\|^2}{2\beta}\right)}\left(\sqrt{s\alpha + s\beta} + O(1)\right) + 8\sqrt{\frac{2\ln(4/\delta)}{m}} \\
=& O\left(L\sqrt{\frac{r\|u_t^*\|^2}{m}\left(2 + \frac{\alpha}{\beta} + \frac{\beta}{\alpha}\right)}\right)
\end{aligned}$$

With the fine-tuning using contrastive learning, in the representation learned, α remains to be 1, while β increases from $1/T$ to $(\sqrt{1/T} + \gamma/\sqrt{s})^2$. Then the error bound decreases. This shows that fine-tuning with contrastive learning on unla-

beled data from the target task can emphasize the task-specific features z_{P_t}, which then leads to better prediction performance.

G APPENDIX FOR CHAPTER 9

G.1 Details of Experimental Setup

G.1.1 Datasets

We use three question answering datasets: CoQA (Reddy et al., 2019), TriviaQA (Joshi et al., 2017) and SQuAD (Rajpurkar et al., 2016) for experiments. The details about these datasets are given below.

CoQA. CoQA is a large-scale dataset for building Conversational Question Answering systems. The goal of the CoQA challenge is to measure the ability of machines to understand a text passage and answer a series of interconnected questions that appear in a conversation. CoQA contains 127,000+ questions with answers collected from 8,000+ conversations. The training set contains 108,647 question queries while the test set contains 7,983 question queries. We use the following template to construct question queries:

```
[The provided context paragraph]
[additional question-answer pairs]
Q: [Provided question]
A:
```

where additional question-answer pairs are preceding turns of the conversation about the paragraph consisting of questions and reference answers.

TriviaQA. TriviaQA is a reading comprehension dataset containing over 650K question-answer-evidence triples. TriviaQA includes 95K question-answer pairs authored by trivia enthusiasts and independently gathered evidence documents, six per question on average, that provide high quality distant supervision for answering the questions. We use TriviaQA as a closed-book QA problem (in which the model must answer a question without access to a supporting paragraph). The training

set contains 138,384 question queries while the test set contains 17,944 question queries. We split the original test set into a new test set containing 8,000 question queries and a validation set containing 9,944 question queries. We use the new test set for evaluation and use the validation set for hyper-parameter selection. We use the following template with a 5-shot prompt to construct question queries:

```
Q: In which decade did Billboard magazine first publish and American
hit chart? A: 30s. Q: What is Bruce Willis' real first name? A:
Walter. Q: Which city does David Soul come from? A: Chicago. Q: Which
William wrote the novel Lord Of The Flies? A: Golding. Q: Where in
England was Dame Judi Dench born? A: York. Q: [Provided question] A:
```

SQuAD. Stanford Question Answering Dataset (SQuAD) is a reading comprehension dataset, consisting of questions posed by crowd-workers on a set of Wikipedia articles, where the answer to every question is a segment of text, or span, from the corresponding reading passage. We use the SQuAD 1.1 version, containing 100,000+ question-answer pairs on 500+ articles. The training set contains 86,821 question queries while the test set contains 5,928 question queries. We use the following template to construct question queries:

```
[The provided context paragraph]
Q: [Provided question]
A:
```

G.1.2 LLMs

We perform experiments with OPT (Zhang et al., 2022) and GPT-2 (Radford et al., 2019) models, which are based on Transformer architecture. For Transformer architecture, there is a limit to the lengths of the sequences we can pass the models. The OPT models can handle sequences of up to 2,048 tokens while the GPT-2 models can handle sequences of up to 1,024 tokens. If the sequence length of an input is larger than the maximum sequence length that is allowed, we force the model to output an empty sequence with a $-\infty$ selection score.

G.1.3 Baselines

For selective prediction, we need to get a predicted output sequence $\hat{\mathbf{y}}^*$ and a selection score $g(\mathbf{x})$ for each input sequence $\mathbf{x}$ given a model f. The model f can be a pre-trained LLM or a LLM adapted via soft prompt tuning using training objective (9.11). We use the beam-search decoding, where the number of beams is equal to 5, to obtain the predicted output sequence $\hat{\mathbf{y}}^*$. We consider the following baselines to compute the selection score $g(\mathbf{x})$:

Perplexity. Perplexity is defined as the exponentiated average negative log-likelihood of a sequence. The perplexity of the generated output sequence $\hat{\mathbf{y}}^*$ is computed as:

$$\text{perp}(\hat{\mathbf{y}}^* \mid \mathbf{x}; f) = f_{\text{norm}}(\hat{\mathbf{y}}^* \mid \mathbf{x})^{-1} \tag{G.1}$$

Predictive Entropy. Predictive entropy is a widely used measure of uncertainty. We use the multinomial sampling with a temperature of 0.5 to obtain an answer list $[\hat{\mathbf{y}}^1, \ldots, \hat{\mathbf{y}}^m]$ for each input sequence $\mathbf{x}$. The predictive entropy is computed as:

$$\text{pe}(\mathbf{x}; f) = \sum_{j=1}^{m} \frac{1}{m} \log f_{\text{norm}}(\hat{\mathbf{y}}^j|\mathbf{x}) \tag{G.2}$$

We set $m = 10$. This is the same as the length-normalised predictive entropy baseline in Kuhn et al. (2023).

Semantic Entropy. Semantic entropy is an entropy-based uncertainty measure which uses a bi-directional entailment algorithm for marginalising over semantically equivalent samples (Kuhn et al., 2023). We follow the settings in Kuhn et al. (2023). Specifically, we use the multinomial sampling with a temperature of 0.5 to obtain an answer list of size 10 for each input sequence for uncertainty estimation. We use the Deberta-large model (He et al., 2020b) that is fine-tuned on the NLI data set, MNLI (Williams et al., 2018) for the bidirectional entailment clustering algorithm.

Self-eval. Self-eval is a simple baseline that obtains a selection score from the LLM by asking whether the proposed answer $\hat{\mathbf{y}}^*$ is correct or not. Suppose $\mathbf{z}_s$ is a series of tokens representing the self-evaluation trigger string "The answer is ". Suppose z_c and z_w are the tokens that represent the words "correct" and "wrong" respectively. Recall that the logits of the model f on v given $\mathbf{x}$ is $\bar{f}(v \mid \mathbf{x})$. Then, the self-eval score is computed as:

$$P(z_c \mid \mathbf{x}, \hat{\mathbf{y}}^*) = \frac{\exp\left(\bar{f}(z_c \mid \mathbf{x}, \hat{\mathbf{y}}^*, \mathbf{z}_s)\right)}{\sum_{z \in \{z_c, z_w\}} \exp\left(\bar{f}(z \mid \mathbf{x}, \hat{\mathbf{y}}^*, \mathbf{z}_s)\right)} \tag{G.3}$$

P(True). P(True) proposed by Kadavath et al. (2022) is a way to estimate the probability that a model's generation is correct by "asking" the model if its answer is correct. It samples m answers and constructs a new natural language question using these possible answers as context before asking whether the proposed answer $\hat{\mathbf{y}}^*$ is correct and measures the probability of the completion being True. We set $m = 4$ and use the multinomial sampling with a temperature of 1.0 to sample the answers. The format of the prompt is:

```
Question: Who was the third president of the United States?
Here are some brainstormed ideas: James Monroe
Thomas Jefferson
John Adams
Benjamin Harrison
George Washington
Possible Answer: James Monroe
Is the possible answer: (A) True (B) False.
The possible answer is:
```

where the "brainstormed answers" are from the set of sampled answers and P(True) (i.e., the likelihood of the next token being True) is taken as the uncertainty measure.

G.1.4 Training Details

We have two stage training: the first stage is to train the soft prompt θ_p using the training objective (9.11) and the second stage is to train the soft prompt θ_s using the training objective (9.12). For both stages, we train the soft prompt for 10 epochs using AdamW optimizer with a batch size of 8, a learning rate of 0.01 and cosine learning rate schedule. We remove those data points $(\mathbf{x}, \mathbf{y})$ where $|\mathbf{x}| + |\mathbf{y}| > 700$ from the training set $\mathcal{D}^{tr}$ to reduce GPU memory usage during training. Here, $|\mathbf{x}|$ is the length of the sequence $\mathbf{x}$. This only removes a very small portion of data points from the training set for each dataset (remove 4.02% training data points in CoQA, 0% training data points in TriviaQA and 0.04% training data points in SQuAD). During training θ_p or θ_s, we always use 20% training data as validation data for selecting the best model among all checkpoints after each training epoch. When training θ_p, we select the best model based on the loss on the validation data. When training θ_s, we select the best model based on the AUROC on the validation data.

G.2 Computational Complexity Analysis

The proposed method ASPIRE needs to train two soft prompts θ_p and θ_s. The complexity of training θ_p using the training objective (9.11) is the same as the complexity of the standard soft prompt tuning. When training θ_s using the training objective (9.12), the number of training steps is the same as that of training θ_p. In each training step of training θ_s, we compute gradients for one correct output and two wrong outputs while in each training step of training θ_p, we compute gradients for one reference output. Thus, the complexity of training θ_s is the same as that of training θ_p. Therefore, the complexity of the proposed method ASPIRE in the training time is the same as that of the standard soft prompt tuning.

We analyze the computational complexity of different methods in the test time in terms of the number of forward passes for the LLM. Since the LLM generates the output sequence in an auto-regressive way, the number of forward passes needed depends on the length of the generated output sequence. Suppose the

maximum length of the generated output sequence is l_{max}. To generate an output sequence given an input sequence, we need one forward pass to encode the input sequence and at most l_{max} forward passes to obtain the output sequence. Thus, for generating the output sequence, the maximum number of forward passes is $1+l_{max}$ and the complexity is $O(l_{max})$. For the perplexity method, the computational complexity is $O(l_{max})$ since we only need additional one forward pass to obtain the perplexity score. For the predictive entropy method, the computational complexity is $O(m \cdot l_{max})$ since we need to additionally generate m answers and compute the likelihood of those m answers. For the semantic entropy method, we omit the computational complexity of the bidirectional entailment clustering algorithm since its computational cost is much smaller than that of the generation of the LLM as stated in Kuhn et al. (2023). Thus, the computational complexity for semantic entropy is $O(m \cdot l_{max})$. For the self-eval method, the computational complexity is $O(l_{max})$ since we only need additional one forward pass to obtain the self-eval score. For the P(True) method, the computational complexity is $O(m \cdot l_{max})$ since we need to additionally generate m answers and need one forward pass to compute the P(True) score. For the proposed method ASPIRE, the computational complexity is $O(l_{max})$ since we only need additional one forward pass to obtain the learned self-eval score. Table G.1 summarizes the computational complexity of different methods in the test time.

Method	Complexity
Perplexity	$O(l_{max})$
Predictive Entropy	$O(m \cdot l_{max})$
Semantic Entropy	$O(m \cdot l_{max})$
Self-eval	$O(l_{max})$
P(True)	$O(m \cdot l_{max})$
ASPIRE (ours)	$O(l_{max})$

Table G.1: Computational complexity of different methods in the test time.

G.3 Rouge Threshold for Evaluation

We use the Rouge-L (Lin and Och, 2004) metric to evaluate if the predicted answer is correct or not. The Rouge-L metric produces a score in $[0, 1]$. We need a threshold γ to determine whether the predicted answer is correct or not. If the Rouge-L score is larger than the threshold γ, then the predicted answer is correct; otherwise, the predicted answer is wrong. The choice of γ depends on the applications. Low values of γ may lead to labeling some wrong answers as correct answers while large values of γ may lead to labeling some correct answers as wrong answers. If we regard the wrong answer as the positive class, then we can use the precision and recall metrics to evaluate the choice of γ. To compute the precision and recall metrics, we need ground-truth labels for determining the correctness of predicted answers, which requires manual labeling. If the Rouge-L score is equal to 0 (or 1), then it is mostly sure that the predicted answer is wrong (or correct). Thus, we only need to label those samples whose Rouge-L scores are in $(0, 1)$. To compare different values of γ, we compute the precision and recall metrics after manually label 200 samples whose Rouge-L scores are in the range of $(0, 1)$. The results in Table G.2 show that larger values of γ lead to higher recall but lower precision while the lower values of γ lead to higher precision but lower recall. In this work, we focus on the safety-critical applications where accepting a wrong answer is more costly compared to rejecting a correct answer that is different from the reference answers. Thus, we prefer high recall than high precision. In our experiments, we evaluate different methods under the Rouge-L metric with $\gamma \in \{0.7, 0.8, 0, 9\}$ to ensure that the recall is at least 90%.

G.4 Rouge Threshold for the Proposed Framework

In the proposed framework ASPIRE, we need the Rouge threshold $\hat{\gamma}$ to determine if the generated answer is correct or not. We want to set a large enough value of $\hat{\gamma}$ so that the generated wrong answers won't be labeled as correct answers. To determine the value of $\hat{\gamma}$, we manually label the correctness of the 10 generated

γ	CoQA		TriviaQA		SQuAD	
	Precision ↑	Recall ↑	Precision ↑	Recall ↑	Precision ↑	Recall ↑
0.1	100.00	0.00	100.00	0.62	100.00	7.91
0.2	100.00	10.00	100.00	2.50	100.00	34.53
0.3	100.00	22.50	100.00	11.88	98.55	48.92
0.4	100.00	45.62	97.01	40.62	93.58	73.38
0.5	97.98	60.62	97.09	62.50	85.94	79.14
0.6	97.41	70.62	96.19	63.12	84.73	79.86
0.7	93.51	90.00	86.81	98.75	76.16	94.24
0.8	86.59	96.88	81.22	100.00	73.66	98.56
0.9	80.71	99.38	80.00	100.00	69.85	100.00

Table G.2: Results of comparing different choices of the Rouge threshold γ. The wrong answer is regarded as the positive class. We use the OPT-2.7B model. We manually label 200 samples with Rouge-L scores in the range of $(0, 1)$ in each dataset and then compute the precision and recall. All numbers are percentages.

answers for 50 training examples from each dataset (we have three datasets CoQA, TriviaQA and SQuAD). The answers are generated using the OPT-2.7B model. We find that if we set $\hat{\gamma} = 0.9$, then no wrong answers will be labeled as correct answers. Thus, we set $\hat{\gamma} = 0.9$ for the proposed framework.

G.5 Complete Results

In this section, we present the complete results for OPT and GPT2 models of different sizes and different Rouge threshold γ. We first evaluate the accuracy of different LLMs. The results are in Table G.3 (set $\gamma = 0.7$), Table G.4 (set $\gamma = 0.8$) and Table G.5 (set $\gamma = 0.9$). The results show that after training θ_p via soft prompt tuning, the accuracy of LLMs is improved significantly. We then evaluate different approaches to compute the selection score when the model's predictions are fixed. The results are in Table G.6 (use GPT2 models and set $\gamma = 0.7$), Table G.7 (use GPT2 models and set $\gamma = 0.8$), Table G.8 (use GPT2 models and set $\gamma = 0.9$), Table G.9 (use OPT models and set $\gamma = 0.7$), Table G.10 (use OPT models and set

$\gamma = 0.8$) and Table G.11 (use OPT models and set $\gamma = 0.9$). The results show that the proposed method ASPIRE significantly outperforms the baselines in terms of AUACC and AUROC across different datasets and LLMs for different values of the Rouge threshold γ.

Model	CoQA	TriviaQA	SQuAD
	Acc ↑	Acc ↑	Acc ↑
Pre-trained GPT2-Medium	35.12	5.44	4.42
Adapted GPT2-Medium with θ_p	57.90	9.04	66.63
Pre-trained GPT2-Large	41.21	8.16	6.09
Adapted GPT2-Large with θ_p	63.89	12.50	71.34
Pre-trained GPT2-XL	46.27	11.80	7.41
Adapted GPT2-XL with θ_p	69.18	17.45	75.44
Pre-trained OPT-350M	28.76	4.35	13.65
Adapted OPT-350M with θ_p	59.46	8.25	64.74
Pre-trained OPT-1.3B	54.13	15.80	30.23
Adapted OPT-1.3B with θ_p	76.85	21.73	80.94
Pre-trained OPT-2.7B	60.68	21.38	35.95
Adapted OPT-2.7B with θ_p	80.45	29.21	83.27

Table G.3: Results of evaluating the accuracy of different LLMs when the Rouge threshold $\gamma = 0.7$. All numbers are percentages.

Model	CoQA	TriviaQA	SQuAD
	Acc ↑	Acc ↑	Acc ↑
Pre-trained GPT2-Medium	32.12	5.00	2.85
Adapted GPT2-Medium with θ_p	55.12	8.71	62.92
Pre-trained GPT2-Large	38.16	7.64	3.98
Adapted GPT2-Large with θ_p	61.04	12.14	67.56
Pre-trained GPT2-XL	42.67	11.10	5.21
Adapted GPT2-XL with θ_p	66.49	16.96	71.17
Pre-trained OPT-350M	27.38	4.25	11.15
Adapted OPT-350M with θ_p	57.02	8.05	61.29
Pre-trained OPT-1.3B	51.35	15.35	25.73
Adapted OPT-1.3B with θ_p	74.46	21.26	77.28
Pre-trained OPT-2.7B	57.72	20.71	30.94
Adapted OPT-2.7B with θ_p	77.97	28.55	80.04

Table G.4: Results of evaluating the accuracy of different LLMs when the Rouge threshold $\gamma = 0.8$. All numbers are percentages.

Model	CoQA	TriviaQA	SQuAD
	Acc ↑	Acc ↑	Acc ↑
Pre-trained GPT2-Medium	30.49	4.88	1.99
Adapted GPT2-Medium with θ_p	53.11	8.53	60.51
Pre-trained GPT2-Large	36.20	7.41	3.00
Adapted GPT2-Large with θ_p	59.04	11.85	64.98
Pre-trained GPT2-XL	40.32	10.82	4.12
Adapted GPT2-XL with θ_p	64.59	16.70	68.83
Pre-trained OPT-350M	26.81	4.20	9.62
Adapted OPT-350M with θ_p	55.33	8.00	59.35
Pre-trained OPT-1.3B	49.78	15.24	22.79
Adapted OPT-1.3B with θ_p	72.78	21.07	74.97
Pre-trained OPT-2.7B	56.06	20.55	27.41
Adapted OPT-2.7B with θ_p	76.45	28.26	78.12

Table G.5: Results of evaluating the accuracy of different LLMs when the Rouge threshold $\gamma = 0.9$. All numbers are percentages.

Model	Method	CoQA		TriviaQA		SQuAD	
		AUACC ↑	AUROC ↑	AUACC ↑	AUROC ↑	AUACC ↑	AUROC ↑
Pre-trained GPT2-Medium	Perplexity	38.92	55.77	7.67	60.52	4.58	55.35
	Predictive Entropy	45.91	62.89	10.02	67.66	5.99	57.07
	Semantic Entropy	48.35	66.30	10.28	68.54	6.18	57.36
	Self-eval	36.16	51.14	6.26	56.74	3.70	43.44
	P(True)	34.08	48.21	8.24	60.62	5.41	54.33
Adapted GPT2-Medium with θ_p	Perplexity	72.03	67.89	18.02	72.17	81.91	72.38
	Predictive Entropy	72.59	69.42	20.07	77.48	82.00	73.09
	Semantic Entropy	73.95	71.54	19.86	77.62	82.35	73.66
	Self-eval	57.94	50.43	9.94	54.68	64.79	46.99
	P(True)	56.71	48.76	13.55	60.79	65.94	49.13
	ASPIRE (ours)	**76.32**	**75.30**	**20.65**	**79.41**	**84.15**	**77.59**
Pre-trained GPT2-Large	Perplexity	48.57	59.82	13.74	66.51	6.39	53.96
	Predictive Entropy	55.04	66.68	16.25	70.46	8.25	57.03
	Semantic Entropy	57.13	69.57	16.02	70.06	8.81	59.24
	Self-eval	42.24	51.72	9.78	54.74	5.07	46.79
	P(True)	36.73	45.69	8.60	48.62	6.83	55.62
Adapted GPT2-Large with θ_p	Perplexity	77.15	68.15	26.83	77.06	86.26	75.34
	Predictive Entropy	77.45	69.76	27.83	80.02	86.32	75.65
	Semantic Entropy	78.85	71.97	27.61	79.88	86.53	75.90
	Self-eval	64.28	50.61	14.26	54.34	70.86	50.81
	P(True)	58.97	45.55	12.38	47.61	70.73	50.09
	ASPIRE (ours)	**81.30**	**76.38**	**29.13**	**82.14**	**87.83**	**79.22**
Pre-trained GPT2-XL	Perplexity	55.93	62.05	22.60	72.88	7.68	51.90
	Predictive Entropy	60.76	67.53	24.83	76.20	10.04	57.21
	Semantic Entropy	63.03	70.50	24.37	75.33	10.38	59.17
	Self-eval	46.67	50.83	9.30	42.75	7.32	49.56
	P(True)	46.98	51.17	10.62	44.54	10.69	60.87
Adapted GPT2-XL with θ_p	Perplexity	83.27	72.79	36.49	79.92	88.73	75.08
	Predictive Entropy	83.49	73.44	37.31	82.21	88.25	74.16
	Semantic Entropy	84.40	75.16	36.68	81.40	88.62	75.26
	Self-eval	69.91	51.90	14.39	43.33	74.26	49.13
	P(True)	70.63	52.83	13.59	40.59	74.34	49.09
	ASPIRE (ours)	**85.65**	**78.32**	**38.06**	**83.23**	**89.86**	**78.35**

Table G.6: Results of evaluating different methods to compute the selection score when the model's predictions are fixed. We use the GPT2 models and set the Rouge threshold $\gamma = 0.7$. All numbers are percentages. **Bold** numbers are superior results.

Model	Method	CoQA		TriviaQA		SQuAD	
		AUACC ↑	AUROC ↑	AUACC ↑	AUROC ↑	AUACC ↑	AUROC ↑
Pre-trained GPT2-Medium	Perplexity	35.24	54.28	7.03	59.54	2.82	53.29
	Predictive Entropy	42.43	62.42	9.23	66.62	3.86	58.15
	Semantic Entropy	45.53	66.84	9.52	67.83	4.02	58.53
	Self-eval	32.97	51.13	5.95	57.98	2.06	40.62
	P(True)	31.05	48.10	7.81	61.51	3.73	55.72
Adapted GPT2-Medium with θ_p	Perplexity	69.36	67.21	17.42	71.77	79.26	72.25
	Predictive Entropy	69.74	68.58	19.38	77.26	79.18	72.70
	Semantic Entropy	71.35	71.10	19.22	77.55	79.61	73.53
	Self-eval	55.04	50.31	9.75	55.26	61.43	47.61
	P(True)	53.95	48.66	13.32	61.55	62.07	49.06
	ASPIRE (ours)	**73.97**	**75.05**	**20.12**	**79.59**	**82.02**	**78.02**
Pre-trained GPT2-Large	Perplexity	44.95	58.70	13.06	66.47	4.06	51.95
	Predictive Entropy	51.57	66.32	15.43	70.33	5.61	57.34
	Semantic Entropy	54.39	70.24	15.25	70.08	6.25	61.09
	Self-eval	39.66	52.36	9.21	54.62	3.15	45.40
	P(True)	33.73	45.72	8.20	49.18	4.68	57.51
Adapted GPT2-Large with θ_p	Perplexity	74.64	67.40	26.20	76.89	83.61	74.57
	Predictive Entropy	74.96	69.07	27.22	80.01	83.57	74.67
	Semantic Entropy	76.65	71.82	27.06	79.99	83.81	75.10
	Self-eval	61.88	50.99	13.83	54.15	67.28	51.20
	P(True)	56.35	45.90	11.95	47.55	67.13	50.52
	ASPIRE (ours)	**79.39**	**76.49**	**28.43**	**82.01**	**85.71**	**79.27**
Pre-trained GPT2-XL	Perplexity	52.07	61.15	21.54	72.72	5.30	49.81
	Predictive Entropy	56.83	66.90	23.65	76.15	7.27	56.53
	Semantic Entropy	59.74	70.83	23.23	75.38	7.59	58.85
	Self-eval	43.34	51.14	8.81	42.76	5.45	51.47
	P(True)	43.24	51.09	9.81	43.94	8.54	65.61
Adapted GPT2-XL with θ_p	Perplexity	81.05	71.85	35.61	79.69	86.08	74.71
	Predictive Entropy	81.23	72.42	36.42	82.01	85.53	73.62
	Semantic Entropy	82.38	74.62	35.84	81.31	85.93	74.84
	Self-eval	67.35	51.89	14.05	43.45	70.30	49.21
	P(True)	68.02	52.83	13.21	40.48	69.47	48.32
	ASPIRE (ours)	**83.91**	**78.09**	**37.26**	**83.18**	**87.76**	**78.82**

Table G.7: Results of evaluating different methods to compute the selection score when the model's predictions are fixed. We use the GPT2 models and set the Rouge threshold $\gamma = 0.8$. All numbers are percentages. **Bold** numbers are superior results.

Model	Method	CoQA		TriviaQA		SQuAD	
		AUACC ↑	AUROC ↑	AUACC ↑	AUROC ↑	AUACC ↑	AUROC ↑
Pre-trained GPT2-Medium	Perplexity	33.09	53.03	6.72	58.75	1.74	47.52
	Predictive Entropy	40.52	62.05	8.90	66.11	2.61	56.43
	Semantic Entropy	44.11	67.38	9.26	67.57	2.85	57.36
	Self-eval	31.46	51.29	5.88	58.50	1.17	35.24
	P(True)	29.24	47.97	7.73	62.30	2.78	58.35
Adapted GPT2-Medium with θ_p	Perplexity	67.42	66.51	17.06	71.50	77.63	72.22
	Predictive Entropy	67.89	68.15	19.06	77.30	77.49	72.55
	Semantic Entropy	69.77	71.20	18.94	77.75	78.07	73.77
	Self-eval	53.15	50.51	9.67	55.80	58.95	47.61
	P(True)	51.95	48.59	13.21	62.06	59.55	48.95
	ASPIRE (ours)	**72.39**	**74.96**	**19.81**	**79.64**	**80.68**	**78.49**
Pre-trained GPT2-Large	Perplexity	42.74	57.93	12.56	65.85	2.78	46.87
	Predictive Entropy	49.68	66.44	14.89	69.76	3.94	54.80
	Semantic Entropy	52.90	71.07	14.76	69.63	4.53	59.75
	Self-eval	38.08	52.84	8.97	54.45	2.42	45.97
	P(True)	31.71	45.48	8.06	49.66	3.84	60.48
Adapted GPT2-Large with θ_p	Perplexity	72.97	66.96	25.67	76.60	81.77	74.01
	Predictive Entropy	73.34	68.78	26.69	79.84	81.80	74.31
	Semantic Entropy	75.24	71.99	26.59	79.96	82.29	75.36
	Self-eval	60.35	51.58	13.44	53.82	64.89	51.42
	P(True)	54.34	45.68	11.65	47.57	64.77	50.75
	ASPIRE (ours)	**78.08**	**76.61**	**27.97**	**81.99**	**84.24**	**79.37**
Pre-trained GPT2-XL	Perplexity	49.71	60.59	20.96	72.31	4.04	46.28
	Predictive Entropy	54.74	67.05	23.10	75.93	5.78	55.59
	Semantic Entropy	58.07	71.83	22.76	75.33	6.18	59.20
	Self-eval	41.19	51.46	8.67	42.97	4.61	53.48
	P(True)	40.37	50.77	9.46	43.58	7.30	69.47
Adapted GPT2-XL with θ_p	Perplexity	79.60	71.40	35.11	79.47	84.69	74.62
	Predictive Entropy	79.86	72.25	35.93	81.85	84.23	73.81
	Semantic Entropy	81.25	74.87	35.39	81.19	84.74	75.38
	Self-eval	65.61	52.08	13.87	43.51	68.10	49.56
	P(True)	65.90	52.61	12.95	40.31	67.37	48.74
	ASPIRE (ours)	**82.77**	**78.11**	**36.81**	**83.09**	**86.57**	**79.06**

Table G.8: Results of evaluating different methods to compute the selection score when the model's predictions are fixed. We use the GPT2 models and set the Rouge threshold $\gamma = 0.9$. All numbers are percentages. **Bold** numbers are superior results.

Model	Method	CoQA		TriviaQA		SQuAD	
		AUACC ↑	AUROC ↑	AUACC ↑	AUROC ↑	AUACC ↑	AUROC ↑
Pre-trained OPT-350M	Perplexity	35.37	59.39	6.81	67.09	13.07	50.34
	Predictive Entropy	36.55	60.31	7.20	65.04	17.86	59.33
	Semantic Entropy	38.80	64.38	7.31	65.15	19.08	61.66
	Self-eval	30.02	52.69	5.98	61.17	14.00	51.41
	P(True)	28.70	50.60	5.29	55.69	17.76	59.55
Adapted OPT-350M with θ_p	Perplexity	74.50	70.21	18.13	75.86	80.64	73.76
	Predictive Entropy	74.14	68.88	18.73	76.83	80.79	73.46
	Semantic Entropy	74.94	70.14	18.46	76.91	81.10	73.98
	Self-eval	60.86	51.67	10.29	57.89	65.48	50.70
	P(True)	59.20	50.04	8.71	52.05	64.55	50.29
	ASPIRE (ours)	**75.55**	**72.37**	**19.00**	**78.54**	**82.59**	**77.18**
Pre-trained OPT-1.3B	Perplexity	69.51	69.32	29.78	74.77	32.43	54.65
	Predictive Entropy	69.46	68.48	31.01	75.21	41.06	62.96
	Semantic Entropy	70.42	70.46	30.63	74.74	43.33	66.30
	Self-eval	56.38	52.86	15.06	49.96	30.74	51.50
	P(True)	57.21	53.19	16.83	51.19	28.88	46.75
Adapted OPT-1.3B with θ_p	Perplexity	88.50	73.64	42.46	79.96	91.45	74.47
	Predictive Entropy	88.24	72.38	43.03	80.46	91.46	74.38
	Semantic Entropy	88.91	74.02	42.70	80.02	91.72	75.44
	Self-eval	78.52	53.08	20.65	49.24	81.05	51.52
	P(True)	79.07	52.76	22.20	50.34	81.58	50.77
	ASPIRE (ours)	**90.76**	**79.26**	**44.03**	**83.06**	**93.41**	**81.17**
Pre-trained OPT-2.7B	Perplexity	75.26	70.16	40.93	78.86	40.82	57.20
	Predictive Entropy	75.29	69.16	41.20	78.92	47.18	62.85
	Semantic Entropy	76.31	70.96	40.72	78.06	51.53	68.40
	Self-eval	62.32	52.26	25.88	59.04	41.78	59.05
	P(True)	62.16	51.80	24.88	56.89	34.77	49.42
Adapted OPT-2.7B with θ_p	Perplexity	90.80	74.23	53.56	81.74	92.86	75.72
	Predictive Entropy	90.63	72.87	53.91	82.19	92.96	75.58
	Semantic Entropy	91.23	74.61	53.58	81.55	93.21	76.53
	Self-eval	81.30	50.76	32.98	56.03	86.34	56.99
	P(True)	81.14	51.01	33.48	56.27	82.59	49.48
	ASPIRE (ours)	**92.63**	**80.25**	**55.06**	**84.44**	**94.73**	**82.60**

Table G.9: Results of evaluating different methods to compute the selection score when the model's predictions are fixed. We use the OPT models and set the Rouge threshold $\gamma = 0.7$. All numbers are percentages. **Bold** numbers are superior results.

Model	Method	CoQA		TriviaQA		SQuAD	
		AUACC ↑	AUROC ↑	AUACC ↑	AUROC ↑	AUACC ↑	AUROC ↑
Pre-trained OPT-350M	Perplexity	33.50	58.50	6.64	66.73	10.50	49.05
	Predictive Entropy	34.88	59.82	7.03	64.84	14.59	58.98
	Semantic Entropy	37.51	64.67	7.17	65.20	15.74	61.85
	Self-eval	28.73	52.91	5.93	61.83	11.33	50.71
	P(True)	27.10	50.36	5.25	56.28	15.38	61.06
Adapted OPT-350M with θ_p	Perplexity	72.04	69.26	17.76	75.70	77.72	72.95
	Predictive Entropy	71.77	68.12	18.30	76.51	77.91	72.76
	Semantic Entropy	72.80	69.90	18.06	76.67	78.35	73.57
	Self-eval	58.65	52.06	10.03	57.87	61.83	50.32
	P(True)	56.82	50.17	8.58	52.27	61.13	50.18
	ASPIRE (ours)	**73.56**	**72.05**	**18.63**	**78.42**	**80.33**	**77.29**
Pre-trained OPT-1.3B	Perplexity	66.09	67.76	29.01	74.46	27.67	53.61
	Predictive Entropy	66.34	67.36	30.21	74.92	35.65	62.44
	Semantic Entropy	67.64	70.02	29.91	74.61	38.00	66.50
	Self-eval	53.87	53.23	14.73	50.12	26.42	51.63
	P(True)	54.07	52.70	16.44	51.38	23.69	45.44
Adapted OPT-1.3B with θ_p	Perplexity	86.67	72.53	41.59	79.61	89.00	73.48
	Predictive Entropy	86.41	71.33	42.18	80.15	89.02	73.35
	Semantic Entropy	87.27	73.41	41.89	79.75	89.42	74.81
	Self-eval	76.49	53.45	20.23	49.20	77.85	52.25
	P(True)	76.79	52.52	21.65	50.25	77.86	50.71
	ASPIRE (ours)	**89.48**	**79.05**	**43.23**	**82.84**	**91.86**	**81.44**
Pre-trained OPT-2.7B	Perplexity	72.00	68.49	39.79	78.43	35.76	56.78
	Predictive Entropy	72.23	67.89	40.05	78.49	41.18	61.98
	Semantic Entropy	73.64	70.43	39.67	77.81	45.83	68.35
	Self-eval	59.51	52.24	25.10	59.02	36.71	59.36
	P(True)	58.81	51.26	24.13	56.80	29.13	48.41
Adapted OPT-2.7B with θ_p	Perplexity	89.10	73.16	52.64	81.56	91.04	74.96
	Predictive Entropy	88.95	72.00	52.97	82.00	91.16	74.86
	Semantic Entropy	89.80	74.53	52.71	81.47	91.46	75.91
	Self-eval	79.12	51.00	32.28	56.03	83.28	56.52
	P(True)	78.74	50.89	32.95	56.42	79.05	49.26
	ASPIRE (ours)	**91.49**	**80.12**	**54.15**	**84.28**	**93.37**	**82.33**

Table G.10: Results of evaluating different methods to compute the selection score when the model's predictions are fixed. We use the OPT models and set the Rouge threshold $\gamma = 0.8$. All numbers are percentages. **Bold** numbers are superior results.

Model	Method	CoQA		TriviaQA		SQuAD	
		AUACC ↑	AUROC ↑	AUACC ↑	AUROC ↑	AUACC ↑	AUROC ↑
Pre-trained OPT-350M	Perplexity	32.58	57.88	6.50	66.50	8.63	46.42
	Predictive Entropy	34.13	59.61	6.91	64.62	12.56	58.33
	Semantic Entropy	36.97	64.81	7.06	65.06	13.82	61.99
	Self-eval	27.98	52.97	5.90	62.10	10.08	51.66
	P(True)	26.41	50.23	5.21	56.37	14.01	62.76
Adapted OPT-350M with θ_p	Perplexity	70.07	68.20	17.67	75.63	76.12	72.40
	Predictive Entropy	69.97	67.41	18.22	76.47	76.44	72.48
	Semantic Entropy	71.38	69.93	17.98	76.65	77.11	73.81
	Self-eval	57.05	52.30	9.96	57.81	59.96	50.56
	P(True)	55.10	50.32	8.53	52.22	59.05	50.03
	ASPIRE (ours)	**71.94**	**71.41**	**18.54**	**78.39**	**79.12**	**77.38**
Pre-trained OPT-1.3B	Perplexity	64.03	66.70	28.77	74.31	24.05	51.41
	Predictive Entropy	64.59	66.80	29.98	74.81	31.35	60.95
	Semantic Entropy	66.29	70.03	29.72	74.56	34.05	66.05
	Self-eval	52.35	53.37	14.64	50.14	24.12	52.63
	P(True)	52.51	52.64	16.27	51.38	20.92	45.41
Adapted OPT-1.3B with θ_p	Perplexity	85.21	71.30	41.21	79.43	87.71	73.17
	Predictive Entropy	85.05	70.44	41.81	80.00	87.81	73.34
	Semantic Entropy	86.23	73.38	41.55	79.66	88.24	74.81
	Self-eval	75.09	53.72	20.07	49.23	75.80	52.60
	P(True)	75.16	52.38	21.44	50.22	75.83	51.10
	ASPIRE (ours)	**88.49**	**78.52**	**42.88**	**82.70**	**90.79**	**81.34**
Pre-trained OPT-2.7B	Perplexity	70.07	67.37	39.42	78.23	31.18	54.43
	Predictive Entropy	70.44	67.03	39.69	78.34	36.14	60.36
	Semantic Entropy	72.29	70.35	39.34	77.68	40.96	67.71
	Self-eval	57.76	52.07	24.85	58.93	32.56	59.52
	P(True)	57.06	50.98	23.96	56.74	25.64	48.02
Adapted OPT-2.7B with θ_p	Perplexity	88.06	72.44	52.12	81.33	90.01	74.59
	Predictive Entropy	87.95	71.48	52.48	81.81	90.17	74.71
	Semantic Entropy	88.96	74.50	52.28	81.35	90.47	75.75
	Self-eval	77.71	51.04	31.90	55.89	81.27	56.36
	P(True)	77.16	50.54	32.62	56.33	76.89	48.85
	ASPIRE (ours)	**90.76**	**79.94**	**53.68**	**84.10**	**92.52**	**82.04**

Table G.11: Results of evaluating different methods to compute the selection score when the model's predictions are fixed. We use the OPT models and set the Rouge threshold $\gamma = 0.9$. All numbers are percentages. **Bold** numbers are superior results.

2021. Online versus batch prediction. https://cloud.google.com/ai-platform/prediction/docs/online-vs-batch-prediction.

Abadi, Martín, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. 2016. Tensorflow: A system for large-scale machine learning. In *12th usenix symposium on operating systems design and implementation (osdi)*, 265–283.

Abdel-Hamid, Ossama, Abdel-rahman Mohamed, Hui Jiang, Li Deng, Gerald Penn, and Dong Yu. 2014. Convolutional neural networks for speech recognition. *IEEE ACM Trans. Audio Speech Lang. Process.* 22(10):1533–1545.

Ajakan, Hana, Pascal Germain, Hugo Larochelle, François Laviolette, and Mario Marchand. 2014. Domain-adversarial neural networks. *stat* 1050:15.

Alayrac, Jean-Baptiste, Jeff Donahue, Pauline Luc, Antoine Miech, Iain Barr, Yana Hasson, Karel Lenc, Arthur Mensch, Katie Millican, Malcolm Reynolds, et al. 2022. Flamingo: a visual language model for few-shot learning. *arXiv preprint arXiv:2204.14198*.

Albuquerque, Isabela, Nikhil Naik, Junnan Li, Nitish Shirish Keskar, and Richard Socher. 2020. Improving out-of-distribution generalization via multi-task self-supervised pretraining. *CoRR* abs/2003.13525. 2003.13525.

Amodei, Dario, Sundaram Ananthanarayanan, Rishita Anubhai, Jingliang Bai, Eric Battenberg, Carl Case, Jared Casper, Bryan Catanzaro, Jingdong Chen, Mike Chrzanowski, Adam Coates, Greg Diamos, Erich Elsen, Jesse H. Engel, Linxi Fan, Christopher Fougner, Awni Y. Hannun, Billy Jun, Tony Han, Patrick LeGresley, Xiangang Li, Libby Lin, Sharan Narang, Andrew Y. Ng, Sherjil Ozair, Ryan Prenger, Sheng Qian, Jonathan Raiman, Sanjeev Satheesh, David Seetapun, Shubho Sengupta, Chong Wang, Yi Wang, Zhiqian Wang, Bo Xiao, Yan Xie, Dani Yogatama,

Jun Zhan, and Zhenyao Zhu. 2016. Deep speech 2 : End-to-end speech recognition in english and mandarin. In *Proceedings of the 33nd international conference on machine learning, ICML 2016, new york city, ny, usa, june 19-24, 2016*, ed. Maria-Florina Balcan and Kilian Q. Weinberger, vol. 48 of *JMLR Workshop and Conference Proceedings*, 173–182. JMLR.org.

Arora, Sanjeev, Hrishikesh Khandeparkar, Mikhail Khodak, Orestis Plevrakis, and Nikunj Saunshi. 2019. A theoretical analysis of contrastive unsupervised representation learning. In *36th international conference on machine learning, icml 2019*, 9904–9923. International Machine Learning Society (IMLS).

Ash, Jordan T, Chicheng Zhang, Akshay Krishnamurthy, John Langford, and Alekh Agarwal. 2019. Deep batch active learning by diverse, uncertain gradient lower bounds. *arXiv preprint arXiv:1906.03671*.

Athalye, Anish, Nicholas Carlini, and David A. Wagner. 2018. Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples. In *Proceedings of the 35th international conference on machine learning (ICML)*, vol. 80 of *Proceedings of Machine Learning Research*, 274–283. PMLR.

Balcan, Maria-Florina, Avrim Blum, Dravyansh Sharma, and Hongyang Zhang. 2023. An analysis of robustness of non-lipschitz networks. *Journal of Machine Learning Research* 24(98):1–43.

Balestriero, Randall, and Yann LeCun. 2022. Contrastive and non-contrastive self-supervised learning recover global and local spectral embedding methods. *arXiv preprint arXiv:2205.11508*.

Barbu, Andrei, David Mayo, Julian Alverio, William Luo, Christopher Wang, Dan Gutfreund, Josh Tenenbaum, and Boris Katz. 2019. Objectnet: A large-scale bias-controlled dataset for pushing the limits of object recognition models. In *Advances in neural information processing systems 32: Annual conference on neural information processing systems 2019, neurips 2019, december 8-14, 2019, vancouver, bc, canada*, ed.

Hanna M. Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d'Alché-Buc, Emily B. Fox, and Roman Garnett, 9448–9458.

Barreno, Marco, Blaine Nelson, Anthony D Joseph, and J Doug Tygar. 2010. The security of machine learning. *Machine Learning* 81(2):121–148.

Barreno, Marco, Blaine Nelson, Russell Sears, Anthony D Joseph, and J Doug Tygar. 2006. Can machine learning be secure? In *Proceedings of the 2006 acm symposium on information, computer and communications security*, 16–25.

Becker, Anja, Léo Ducas, Nicolas Gama, and Thijs Laarhoven. 2016. New directions in nearest neighbor searching with applications to lattice sieving. In *Proceedings of the twenty-seventh annual acm-siam symposium on discrete algorithms*, 10–24. SIAM.

Beery, Sara, Elijah Cole, and Arvi Gjoka. 2020. The iwildcam 2020 competition dataset. *arXiv preprint arXiv:2004.10340*.

Beluch, William H, Tim Genewein, Andreas Nürnberger, and Jan M Köhler. 2018. The power of ensembles for active learning in image classification. In *Proceedings of the ieee conference on computer vision and pattern recognition*, 9368–9377.

Ben-Tal, A., L. El Ghaoui, and A.S. Nemirovski. 2009. *Robust optimization*. Princeton Series in Applied Mathematics, Princeton University Press.

Bendale, Abhijit, and Terrance Boult. 2015. Towards open world recognition. In *Cvpr*, 1893–1902.

Benjamin Bossan, Wendy Kan, Josef Feigl. 2015. Otto group product classification challenge.

Bevandić, Petra, Ivan Krešo, Marin Oršić, and Siniša Šegvić. 2018. Discriminative out-of-distribution detection for semantic segmentation. *arXiv preprint arXiv:1808.07703*.

Biggio, Battista, Igino Corona, Davide Maiorca, Blaine Nelson, Nedim Šrndić, Pavel Laskov, Giorgio Giacinto, and Fabio Roli. 2013a. Evasion attacks against

machine learning at test time. In *Joint european conference on machine learning and knowledge discovery in databases*, 387–402. Springer.

Biggio, Battista, Igino Corona, Davide Maiorca, Blaine Nelson, Nedim Srndic, Pavel Laskov, Giorgio Giacinto, and Fabio Roli. 2013b. Evasion attacks against machine learning at test time. In *Machine learning and knowledge discovery in databases - european conference, ECML PKDD 2013, prague, czech republic, september 23-27, 2013, proceedings, part III*, ed. Hendrik Blockeel, Kristian Kersting, Siegfried Nijssen, and Filip Zelezný, vol. 8190 of *Lecture Notes in Computer Science*, 387–402. Springer.

Biggio, Battista, Giorgio Fumera, and Fabio Roli. 2010. Multiple classifier systems for robust classifier design in adversarial environments. *International Journal of Machine Learning and Cybernetics* 1(1):27–41.

Biggio, Battista, and Fabio Roli. 2018. Wild patterns: Ten years after the rise of adversarial machine learning. *Pattern Recognition* 84:317–331.

Bitterwolf, Julian, Alexander Meinke, and Matthias Hein. 2020. Certifiably adversarially robust detection of out-of-distribution data. *NeurIPS* 33.

Blitzer, John, Mark Dredze, and Fernando Pereira. 2007. Domain adaptation for sentiment classification. In *45th annv. meeting of the assoc. computational linguistics (acl'07)*.

Bommasani, Rishi, Drew A. Hudson, Ehsan Adeli, Russ Altman, Simran Arora, Sydney von Arx, Michael S. Bernstein, Jeannette Bohg, Antoine Bosselut, Emma Brunskill, Erik Brynjolfsson, Shyamal Buch, Dallas Card, Rodrigo Castellon, Niladri S. Chatterji, Annie S. Chen, Kathleen Creel, Jared Quincy Davis, Dorottya Demszky, Chris Donahue, Moussa Doumbouya, Esin Durmus, Stefano Ermon, John Etchemendy, Kawin Ethayarajh, Li Fei-Fei, Chelsea Finn, Trevor Gale, Lauren Gillespie, Karan Goel, Noah D. Goodman, Shelby Grossman, Neel Guha, Tatsunori Hashimoto, Peter Henderson, John Hewitt, Daniel E. Ho, Jenny Hong, Kyle Hsu, Jing Huang, Thomas Icard, Saahil Jain, Dan Jurafsky, Pratyusha Kalluri, Siddharth Karamcheti, Geoff Keeling, Fereshte Khani, Omar Khattab, Pang Wei Koh, Mark S.

Krass, Ranjay Krishna, Rohith Kuditipudi, and et al. 2021. On the opportunities and risks of foundation models. *CoRR* abs/2108.07258. 2108.07258.

Breunig, Markus M, Hans-Peter Kriegel, Raymond T Ng, and Jörg Sander. 2000. Lof: identifying density-based local outliers. In *Proceedings of the 2000 acm sigmod international conference on management of data*, 93–104.

Brown, Tom B., Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. In *Advances in neural information processing systems 33: Annual conference on neural information processing systems 2020, neurips 2020, december 6-12, 2020, virtual*, ed. Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin.

Carlini, Nicholas, Anish Athalye, Nicolas Papernot, Wieland Brendel, Jonas Rauber, Dimitris Tsipras, Ian Goodfellow, Aleksander Madry, and Alexey Kurakin. 2019. On evaluating adversarial robustness. *arXiv preprint arXiv:1902.06705*.

Carlini, Nicholas, and David A. Wagner. 2017. Towards evaluating the robustness of neural networks. In *2017 IEEE symposium on security and privacy, SP 2017, san jose, ca, usa, may 22-26, 2017*, 39–57. IEEE Computer Society.

Carmon, Yair, Aditi Raghunathan, Ludwig Schmidt, John C. Duchi, and Percy Liang. 2019. Unlabeled data improves adversarial robustness. In *Advances in neural information processing systems 32: Annual conference on neural information processing systems 2019, neurips 2019, 8-14 december 2019, vancouver, bc, canada*, ed. Hanna M. Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d'Alché-Buc, Emily B. Fox, and Roman Garnett, 11190–11201.

Charoenphakdee, Nontawat, Zhenghang Cui, Yivan Zhang, and Masashi Sugiyama. 2021. Classification with rejection based on cost-sensitive classification. In *Proceedings of the 38th international conference on machine learning (ICML)*, vol. 139 of *Proceedings of Machine Learning Research*, 1507–1517. PMLR.

Chen, Jiefeng, Yixuan Li, Xi Wu, Yingyu Liang, and Somesh Jha. 2021a. ATOM: robustifying out-of-distribution detection using outlier mining. In *Machine learning and knowledge discovery in databases. research track - european conference, ECML PKDD 2021, bilbao, spain, september 13-17, 2021, proceedings, part III*, ed. Nuria Oliver, Fernando Pérez-Cruz, Stefan Kramer, Jesse Read, and José Antonio Lozano, vol. 12977 of *Lecture Notes in Computer Science*, 430–445. Springer.

Chen, Jiefeng, Frederick Liu, Besim Avci, Xi Wu, Yingyu Liang, and Somesh Jha. 2021b. Detecting errors and estimating accuracy on unlabeled data with self-training ensembles. *Advances in Neural Information Processing Systems* 34.

Chen, Jiefeng, Jayaram Raghuram, Jihye Choi, Xi Wu, Yingyu Liang, and Somesh Jha. 2023a. Stratified adversarial robustness with rejection. In *International conference on machine learning*.

Chen, Jiefeng, Xi Wu, Yang Guo, Yingyu Liang, and Somesh Jha. 2021c. Towards evaluating the robustness of neural networks learned by transduction. In *International conference on learning representations*.

Chen, Jiefeng, Xi Wu, Vaibhav Rastogi, Yingyu Liang, and Somesh Jha. 2019. Robust attribution regularization. In *Advances in neural information processing systems 32: Annual conference on neural information processing systems 2019, neurips 2019, december 8-14, 2019, vancouver, bc, canada*, ed. Hanna M. Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d'Alché-Buc, Emily B. Fox, and Roman Garnett, 14300–14310.

Chen, Jiefeng, Jinsung Yoon, Sayna Ebrahimi, Sercan Arik, Somesh Jha, and Tomas Pfister. 2023b. Aspest: Bridging the gap between active learning and selective prediction. *arXiv preprint arXiv:2304.03870*.

Chen, Liang-Chieh, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L. Yuille. 2018. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *IEEE Trans. Pattern Anal. Mach. Intell.* 40(4):834–848.

Chen, Mayee F., Karan Goel, Nimit Sharad Sohoni, Fait Poms, Kayvon Fatahalian, and Christopher Ré. 2021d. Mandoline: Model evaluation under distribution shift. In *Proceedings of the 38th international conference on machine learning, ICML 2021, 18-24 july 2021, virtual event*, ed. Marina Meila and Tong Zhang, vol. 139 of *Proceedings of Machine Learning Research*, 1617–1629. PMLR.

Chen, Ting, Simon Kornblith, Mohammad Norouzi, and Geoffrey E. Hinton. 2020a. A simple framework for contrastive learning of visual representations. In *Proceedings of the 37th international conference on machine learning, ICML 2020, 13-18 july 2020, virtual event*, vol. 119 of *Proceedings of Machine Learning Research*, 1597–1607. PMLR.

Chen, Ting, Simon Kornblith, Kevin Swersky, Mohammad Norouzi, and Geoffrey E. Hinton. 2020b. Big self-supervised models are strong semi-supervised learners. In *Advances in neural information processing systems 33: Annual conference on neural information processing systems 2020, neurips 2020, december 6-12, 2020, virtual*, ed. Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin.

Chen, Xinlei, and Kaiming He. 2021. Exploring simple siamese representation learning. In *IEEE conference on computer vision and pattern recognition, CVPR 2021, virtual, june 19-25, 2021*, 15750–15758. Computer Vision Foundation / IEEE.

Chorowski, Jan, Dzmitry Bahdanau, Dmitriy Serdyuk, Kyunghyun Cho, and Yoshua Bengio. 2015. Attention-based models for speech recognition. In *Advances in neural information processing systems 28: Annual conference on neural information processing systems 2015, december 7-12, 2015, montreal, quebec, canada*, ed. Corinna Cortes, Neil D. Lawrence, Daniel D. Lee, Masashi Sugiyama, and Roman Garnett, 577–585.

Chowdhery, Aakanksha, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, et al. 2022. Palm: Scaling language modeling with pathways. *arXiv preprint arXiv:2204.02311*.

Chrabaszcz, Patryk, Ilya Loshchilov, and Frank Hutter. 2017. A downsampled variant of imagenet as an alternative to the cifar datasets. *arXiv preprint arXiv:1707.08819*.

Christie, Gordon, Neil Fendley, James Wilson, and Ryan Mukherjee. 2018. Functional map of the world. In *Proceedings of the ieee conference on computer vision and pattern recognition*, 6172–6180.

Chuang, Ching-Yao, Antonio Torralba, and Stefanie Jegelka. 2020. Estimating generalization under distribution shifts via domain-invariant representations. *arXiv preprint arXiv:2007.03511*.

Cimpoi, M., S. Maji, I. Kokkinos, S. Mohamed, , and A. Vedaldi. 2014. Describing textures in the wild. In *Cvpr*.

Cohen, Jeremy M., Elan Rosenfeld, and J. Zico Kolter. 2019. Certified adversarial robustness via randomized smoothing. In *Proceedings of the 36th international conference on machine learning, ICML 2019, 9-15 june 2019, long beach, california, USA*, ed. Kamalika Chaudhuri and Ruslan Salakhutdinov, vol. 97 of *Proceedings of Machine Learning Research*, 1310–1320. PMLR.

Cole, Elijah, Xuan Yang, Kimberly Wilber, Oisin Mac Aodha, and Serge Belongie. 2022. When does contrastive visual representation learning work? In *Proceedings of the ieee/cvf conference on computer vision and pattern recognition*, 14755–14764.

Cole, Jeremy R, Michael JQ Zhang, Daniel Gillick, Julian Martin Eisenschlos, Bhuwan Dhingra, and Jacob Eisenstein. 2023. Selectively answering ambiguous questions. *arXiv preprint arXiv:2305.14613*.

Colson, Benoît, Patrice Marcotte, and Gilles Savard. 2007. An overview of bilevel optimization.

Corbière, Charles, Nicolas Thome, Avner Bar-Hen, Matthieu Cord, and Patrick Pérez. 2019. Addressing failure prediction by learning model confidence. In *Advances in neural information processing systems*, 2902–2913.

Cortes, Corinna, Giulia DeSalvo, and Mehryar Mohri. 2016. Learning with rejection. In *Algorithmic learning theory - 27th international conference, ALT*, vol. 9925 of *Lecture Notes in Computer Science*, 67–82.

Croce, Francesco, Maksym Andriushchenko, Vikash Sehwag, Nicolas Flammarion, Mung Chiang, Prateek Mittal, and Matthias Hein. 2020. Robustbench: a standardized adversarial robustness benchmark. *CoRR* abs/2010.09670. 2010.09670.

Croce, Francesco, and Matthias Hein. 2020. Reliable evaluation of adversarial robustness with an ensemble of diverse parameter-free attacks. In *Proceedings of the 37th international conference on machine learning, ICML 2020, 13-18 july 2020, virtual event*, vol. 119 of *Proceedings of Machine Learning Research*, 2206–2216. PMLR.

Cui, Yin, Feng Zhou, Yuanqing Lin, and Serge Belongie. 2016. Fine-grained categorization and dataset bootstrapping using deep metric learning with humans in the loop. In *Cvpr*, 1153–1162.

Cunningham, Mitchell, and Michael A Regan. 2015. Autonomous vehicles: human factors issues and future research. In *Proceedings of the 2015 australasian road safety conference*, vol. 14.

Dalvi, Nilesh, Pedro Domingos, Sumit Sanghai, and Deepak Verma. 2004. Adversarial classification. In *Proceedings of the tenth acm sigkdd international conference on knowledge discovery and data mining*, 99–108.

Darlow, Luke N, Elliot J Crowley, Antreas Antoniou, and Amos J Storkey. 2018. Cinic-10 is not imagenet or cifar-10. *arXiv preprint arXiv:1810.03505*.

Dasgupta, Sanjoy. 2011. Two faces of active learning. *Theoretical computer science* 412(19):1767–1781.

Deng, Weijian, Stephen Gould, and Liang Zheng. 2021. What does rotation prediction tell us about classifier accuracy under varying testing environments? In *Proceedings of the 38th international conference on machine learning, ICML 2021, 18-24 july 2021, virtual event*, ed. Marina Meila and Tong Zhang, vol. 139 of *Proceedings of Machine Learning Research*, 2579–2589. PMLR.

Deng, Weijian, and Liang Zheng. 2021. Are labels always necessary for classifier accuracy evaluation? In *IEEE conference on computer vision and pattern recognition, CVPR 2021, virtual, june 19-25, 2021*, 15069–15078. Computer Vision Foundation / IEEE.

Devlin, Jacob, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 conference of the north american chapter of the association for computational linguistics: Human language technologies, NAACL-HLT 2019, minneapolis, mn, usa, june 2-7, 2019, volume 1 (long and short papers)*, ed. Jill Burstein, Christy Doran, and Thamar Solorio, 4171–4186. Association for Computational Linguistics.

Ding, Gavin Weiguang, Yash Sharma, Kry Yik Chau Lui, and Ruitong Huang. 2020. MMA training: Direct input space margin maximization through adversarial training. In *8th international conference on learning representations, ICLR 2020, addis ababa, ethiopia, april 26-30, 2020*. OpenReview.net.

Ding, Jianbang, Xuancheng Ren, Ruixuan Luo, and Xu Sun. 2019. An adaptive and momental bound method for stochastic learning. *CoRR* abs/1910.12249. 1910.12249.

Doersch, Carl, Abhinav Gupta, and Alexei A. Efros. 2015. Unsupervised visual representation learning by context prediction. In *2015 IEEE international conference on computer vision, ICCV 2015, santiago, chile, december 7-13, 2015*, 1422–1430. IEEE Computer Society.

Dolan, William B., and Chris Brockett. 2005. Automatically constructing a corpus of sentential paraphrases. In *Proceedings of the third international workshop on paraphrasing, iwp@ijcnlp 2005, jeju island, korea, october 2005, 2005*. Asian Federation of Natural Language Processing.

Donmez, Pinar, Guy Lebanon, and Krishnakumar Balasubramanian. 2010. Unsupervised supervised learning i: Estimating classification and regression errors without labels. *Journal of Machine Learning Research* 11(4).

Dosovitskiy, Alexey, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. 2020. An image is worth 16x16 words: Transformers for image recognition at scale. In *International conference on learning representations*.

Drucker, Harris, and Yann LeCun. 1992. Improving generalization performance using double backpropagation. *IEEE Trans. Neural Networks* 3(6):991–997.

Duan, Yueqi, Lei Chen, Jiwen Lu, and Jie Zhou. 2019. Deep embedding learning with discriminative sampling policy. In *Cvpr*, 4964–4973.

Ducoffe, Melanie, and Frederic Precioso. 2018. Adversarial active learning for deep networks: a margin based approach. *arXiv preprint arXiv:1802.09841*.

El-Yaniv, Ran, et al. 2010. On the foundations of noise-free selective classification. *Journal of Machine Learning Research* 11(5).

Elsahar, Hady, and Matthias Gallé. 2019. To annotate or not? predicting performance drop under domain shift. In *Proceedings of the 2019 conference on empirical methods in natural language processing and the 9th international joint conference on natural language processing (emnlp-ijcnlp)*, 2163–2173.

Engstrom, Logan, Andrew Ilyas, Shibani Santurkar, Dimitris Tsipras, Brandon Tran, and Aleksander Madry. 2019. Adversarial robustness as a prior for learned representations. *arXiv 1906.00945*.

Ericsson, Linus, Henry Gouk, and Timothy M. Hospedales. 2021. How well do self-supervised models transfer? In *IEEE conference on computer vision and pattern recognition, CVPR 2021, virtual, june 19-25, 2021*, 5414–5423. Computer Vision Foundation / IEEE.

Eykholt, Kevin, Ivan Evtimov, Earlence Fernandes, Bo Li, Amir Rahmati, Chaowei Xiao, Atul Prakash, Tadayoshi Kohno, and Dawn Song. 2018. Robust physical-world attacks on deep learning visual classification. In *Proceedings of the ieee conference on computer vision and pattern recognition*, 1625–1634.

Felzenszwalb, Pedro F, Ross B Girshick, David McAllester, and Deva Ramanan. 2009. Object detection with discriminatively trained part-based models. *IEEE transactions on pattern analysis and machine intelligence* 32(9):1627–1645.

Filos, Angelos, Panagiotis Tigkas, Rowan McAllister, Nicholas Rhinehart, Sergey Levine, and Yarin Gal. 2020. Can autonomous vehicles identify, recover from, and adapt to distribution shifts? In *Icml*, 3145–3153. PMLR.

Fort, Stanislav, Huiyi Hu, and Balaji Lakshminarayanan. 2019. Deep ensembles: A loss landscape perspective. *arXiv preprint arXiv:1912.02757*.

Fu, Bo, Zhangjie Cao, Jianmin Wang, and Mingsheng Long. 2021. Transferable query selection for active domain adaptation. In *Proceedings of the ieee/cvf conference on computer vision and pattern recognition*, 7272–7281.

Fumera, Giorgio, and Fabio Roli. 2002. Support vector machines with embedded reject option. In *International workshop on support vector machines*, 68–82. Springer.

Gal, Yarin, and Zoubin Ghahramani. 2016. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *international conference on machine learning*, 1050–1059.

Gal, Yarin, Riashat Islam, and Zoubin Ghahramani. 2017. Deep bayesian active learning with image data. In *International conference on machine learning*, 1183–1192. PMLR.

Ganin, Yaroslav, Evgeniya Ustinova, Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, Mario Marchand, and Victor Lempitsky. 2016. Domain-adversarial training of neural networks. *The journal of machine learning research* 17(1):2096–2030.

Gao, Tianyu, Xingcheng Yao, and Danqi Chen. 2021. Simcse: Simple contrastive learning of sentence embeddings. In *Proceedings of the 2021 conference on empirical methods in natural language processing*, 6894–6910.

Garg, Siddhant, and Yingyu Liang. 2020. Functional regularization for representation learning: A unified theoretical perspective. *Advances in Neural Information Processing Systems* 33:17187–17199.

Geifman, Yonatan, and Ran El-Yaniv. 2017. Selective classification for deep neural networks. *Advances in neural information processing systems* 30.

Geifman, Yonatan, and Ran El-Yaniv. 2019. SelectiveNet: A deep neural network with an integrated reject option. In *Proceedings of the 36th international conference on machine learning (ICML)*, vol. 97 of *Proceedings of Machine Learning Research*, 2151–2159. PMLR.

Ghorbani, Amirata, Abubakar Abid, and James Y. Zou. 2019. Interpretation of neural networks is fragile. In *The thirty-third AAAI conference on artificial intelligence, AAAI 2019, the thirty-first innovative applications of artificial intelligence conference, IAAI 2019, the ninth AAAI symposium on educational advances in artificial intelligence, EAAI 2019, honolulu, hawaii, usa, january 27 - february 1, 2019*, 3681–3688. AAAI Press.

Gidaris, Spyros, and Nikos Komodakis. 2015. Object detection via a multi-region and semantic segmentation-aware cnn model. In *Iccv*, 1134–1142.

Globerson, Amir, and Sam Roweis. 2006. Nightmare at test time: robust learning by feature deletion. In *Proceedings of the 23rd international conference on machine learning*, 353–360.

Goldwasser, Shafi, Adam Tauman Kalai, Yael Tauman Kalai, and Omar Montasser. 2020. Beyond perturbations: Learning guarantees with arbitrary adversarial test examples. *CoRR* abs/2007.05145. 2007.05145.

Goodfellow, Ian, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative adversarial nets. *Advances in neural information processing systems* 27.

Goodfellow, Ian J, Jonathon Shlens, and Christian Szegedy. 2015. Explaining and harnessing adversarial examples. *ICLR*.

Grandvalet, Yves, and Yoshua Bengio. 2004. Semi-supervised learning by entropy minimization. *Advances in neural information processing systems* 17.

Granese, Federica, Marco Romanelli, Daniele Gorla, Catuscia Palamidessi, and Pablo Piantanida. 2021. Doctor: A simple method for detecting misclassification errors. *Advances in Neural Information Processing Systems* 34:5669–5681.

Grill, Jean-Bastien, Florian Strub, Florent Altché, Corentin Tallec, Pierre H. Richemond, Elena Buchatskaya, Carl Doersch, Bernardo Ávila Pires, Zhaohan Guo, Mohammad Gheshlaghi Azar, Bilal Piot, Koray Kavukcuoglu, Rémi Munos, and Michal Valko. 2020. Bootstrap your own latent - A new approach to self-supervised learning. In *Advances in neural information processing systems 33: Annual conference on neural information processing systems 2020, neurips 2020, december 6-12, 2020, virtual*.

Guillory, Devin, Vaishaal Shankar, Sayna Ebrahimi, Trevor Darrell, and Ludwig Schmidt. 2021. Predicting with confidence on unseen distributions. *CoRR* abs/2107.03315. 2107.03315.

Guo, Chuan, Geoff Pleiss, Yu Sun, and Kilian Q Weinberger. 2017. On calibration of modern neural networks. *arXiv preprint arXiv:1706.04599*.

Hacohen, Guy, Avihu Dekel, and Daphna Weinshall. 2022. Active learning on a budget: Opposite strategies suit high and low budgets. *arXiv preprint arXiv:2202.02794*.

Hannun, Awni Y., Carl Case, Jared Casper, Bryan Catanzaro, Greg Diamos, Erich Elsen, Ryan Prenger, Sanjeev Satheesh, Shubho Sengupta, Adam Coates, and Andrew Y. Ng. 2014. Deep speech: Scaling up end-to-end speech recognition. *CoRR* abs/1412.5567. `1412.5567`.

HaoChen, Jeff Z, Colin Wei, Adrien Gaidon, and Tengyu Ma. 2021. Provable guarantees for self-supervised deep learning with spectral contrastive loss. *Advances in Neural Information Processing Systems* 34.

Harwood, Ben, Vijay Kumar BG, Gustavo Carneiro, Ian Reid, and Tom Drummond. 2017. Smart mining for deep metric learning. In *Iccv*, 2821–2829.

He, Kaiming, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick. 2022. Masked autoencoders are scalable vision learners. In *Proceedings of the ieee/cvf conference on computer vision and pattern recognition*, 16000–16009.

He, Kaiming, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross B. Girshick. 2020a. Momentum contrast for unsupervised visual representation learning. In *2020 IEEE/CVF conference on computer vision and pattern recognition, CVPR 2020, seattle, wa, usa, june 13-19, 2020*, 9726–9735. Computer Vision Foundation / IEEE.

He, Kaiming, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016a. Deep residual learning for image recognition. In *2016 IEEE conference on computer vision and pattern recognition, CVPR 2016, las vegas, nv, usa, june 27-30, 2016*, 770–778. IEEE Computer Society.

———. 2016b. Identity mappings in deep residual networks. In *European conference on computer vision*, 630–645. Springer.

He, Pengcheng, Xiaodong Liu, Jianfeng Gao, and Weizhu Chen. 2020b. Deberta: Decoding-enhanced bert with disentangled attention. *arXiv preprint arXiv:2006.03654*.

Hein, Matthias, Maksym Andriushchenko, and Julian Bitterwolf. 2019. Why relu networks yield high-confidence predictions far away from the training data and how to mitigate the problem. In *IEEE conference on computer vision and pattern recognition, CVPR 2019, long beach, ca, usa, june 16-20, 2019*, 41–50. Computer Vision Foundation / IEEE.

Hellman, Martin E. 1970. The nearest neighbor classification rule with a reject option. *IEEE Transactions on Systems Science and Cybernetics* 6(3):179–185.

Hendrycks, Dan, and Thomas G. Dietterich. 2019. Benchmarking neural network robustness to common corruptions and perturbations. In *7th international conference on learning representations, ICLR 2019, new orleans, la, usa, may 6-9, 2019*. OpenReview.net.

Hendrycks, Dan, and Kevin Gimpel. 2017. A baseline for detecting misclassified and out-of-distribution examples in neural networks. *ICLR*.

Hendrycks, Dan, Kimin Lee, and Mantas Mazeika. 2019a. Using pre-training can improve model robustness and uncertainty. In *Proceedings of the 36th international conference on machine learning, ICML 2019, 9-15 june 2019, long beach, california, USA*, ed. Kamalika Chaudhuri and Ruslan Salakhutdinov, vol. 97 of *Proceedings of Machine Learning Research*, 2712–2721. PMLR.

Hendrycks, Dan, Mantas Mazeika, and Thomas G. Dietterich. 2019b. Deep anomaly detection with outlier exposure. In *7th international conference on learning representations, ICLR 2019, new orleans, la, usa, may 6-9, 2019*. OpenReview.net.

Hotelling, Harold. 1933. Analysis of a complex of statistical variables into principal components. *Journal of educational psychology* 24(6):417.

Howard, Jeremy, and Sebastian Ruder. 2018. Universal language model fine-tuning for text classification. In *Proceedings of the 56th annual meeting of the association for computational linguistics, ACL 2018, melbourne, australia, july 15-20, 2018, volume 1: Long papers*, ed. Iryna Gurevych and Yusuke Miyao, 328–339. Association for Computational Linguistics.

Hsu, Yen-Chang, Yilin Shen, Hongxia Jin, and Zsolt Kira. 2020. Generalized odin: Detecting out-of-distribution image without learning from out-of-distribution data. *CVPR*.

Hu, Edward J, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, Weizhu Chen, et al. 2021. Lora: Low-rank adaptation of large language models. In *International conference on learning representations*.

Huang, Gao, Yixuan Li, Geoff Pleiss, Zhuang Liu, John E Hopcroft, and Kilian Q Weinberger. 2017a. Snapshot ensembles: Train 1, get m for free. *arXiv preprint arXiv:1704.00109*.

Huang, Gao, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. 2017b. Densely connected convolutional networks. In *Proceedings of the ieee conference on computer vision and pattern recognition*, 4700–4708.

Huang, Rui, and Yixuan Li. 2021. Towards scaling out-of-distribution detection for large semantic space. *CVPR*.

Huang, Sheng-Jun, Rong Jin, and Zhi-Hua Zhou. 2010. Active learning by querying informative and representative examples. *Advances in neural information processing systems* 23.

Hull, Jonathan J. 1994. A database for handwritten text recognition research. *IEEE Transactions on pattern analysis and machine intelligence* 16(5):550–554.

Jacot, Arthur, Franck Gabriel, and Clement Hongler. 2018. Neural tangent kernel: Convergence and generalization in neural networks. In *Advances in neural infor-*

mation processing systems, ed. S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, vol. 31. Curran Associates, Inc.

Jaffe, Ariel, Boaz Nadler, and Yuval Kluger. 2015. Estimating the accuracies of multiple classifiers without labeled data. In *Proceedings of the eighteenth international conference on artificial intelligence and statistics, AISTATS 2015, san diego, california, usa, may 9-12, 2015*, ed. Guy Lebanon and S. V. N. Vishwanathan, vol. 38 of *JMLR Workshop and Conference Proceedings*. JMLR.org.

Jiang, Heinrich, Been Kim, Melody Guan, and Maya Gupta. 2018. To trust or not to trust a classifier. In *Advances in neural information processing systems*, 5541–5552.

Jiang, Yiding, Vaishnavh Nagarajan, Christina Baek, and J. Zico Kolter. 2021a. Assessing generalization of SGD via disagreement. *CoRR* abs/2106.13799. 2106.13799.

Jiang, Zhengbao, Jun Araki, Haibo Ding, and Graham Neubig. 2021b. How can we know when language models know? on the calibration of language models for question answering. *Transactions of the Association for Computational Linguistics* 9: 962–977.

Johansson, Fredrik D, David Sontag, and Rajesh Ranganath. 2019. Support and invertibility in domain-invariant representations. In *The 22nd international conference on artificial intelligence and statistics*, 527–536. PMLR.

Joshi, Mandar, Eunsol Choi, Daniel S Weld, and Luke Zettlemoyer. 2017. Triviaqa: A large scale distantly supervised challenge dataset for reading comprehension. *arXiv preprint arXiv:1705.03551*.

Kadavath, Saurav, Tom Conerly, Amanda Askell, Tom Henighan, Dawn Drain, Ethan Perez, Nicholas Schiefer, Zac Hatfield Dodds, Nova DasSarma, Eli Tran-Johnson, et al. 2022. Language models (mostly) know what they know. *arXiv preprint arXiv:2207.05221*.

Kalibhat, Neha Mukund, Kanika Narang, Hamed Firooz, Maziar Sanjabi, and Soheil Feizi. 2022. Towards better understanding of self-supervised representations. In *Icml 2022: Workshop on spurious correlations, invariance and stability*.

Kamath, Amita, Robin Jia, and Percy Liang. 2020. Selective question answering under domain shift. *arXiv preprint arXiv:2006.09462*.

Kato, Masahiro, Zhenghang Cui, and Yoshihiro Fukuhara. 2020. ATRO: adversarial training with a rejection option. *CoRR* abs/2010.12905. 2010.12905.

Kendall, Maurice G. 1938. A new measure of rank correlation. *Biometrika* 30(1/2): 81–93.

Kim, Daehee, Youngjun Yoo, Seunghyun Park, Jinkyu Kim, and Jaekoo Lee. 2021. Selfreg: Self-supervised contrastive regularization for domain generalization. In *Proceedings of the ieee/cvf international conference on computer vision*, 9619–9628.

Kingma, Diederik P, and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Kirsch, Andreas, Tom Rainforth, and Yarin Gal. 2021. Test distribution-aware active learning: A principled approach against distribution shift and outliers. *arXiv preprint arXiv:2106.11719*.

Ko, Ching-Yun, Jeet Mohapatra, Sijia Liu, Pin-Yu Chen, Luca Daniel, and Lily Weng. 2022. Revisiting contrastive learning through the lens of neighborhood component analysis: an integrated framework. In *International conference on machine learning*, 11387–11412. PMLR.

Kobayashi, Sosuke, Shun Kiyono, Jun Suzuki, and Kentaro Inui. 2022. Diverse lottery tickets boost ensemble from a single pretrained model. *arXiv preprint arXiv:2205.11833*.

Koh, Pang Wei, Shiori Sagawa, Henrik Marklund, Sang Michael Xie, Marvin Zhang, Akshay Balsubramani, Weihua Hu, Michihiro Yasunaga, Richard Lanas Phillips,

Irena Gao, et al. 2021. Wilds: A benchmark of in-the-wild distribution shifts. In *International conference on machine learning*, 5637–5664. PMLR.

Kolesnikov, Alexander, Lucas Beyer, Xiaohua Zhai, Joan Puigcerver, Jessica Yung, Sylvain Gelly, and Neil Houlsby. 2020. Big transfer (bit): General visual representation learning. In *Computer vision - ECCV 2020 - 16th european conference, glasgow, uk, august 23-28, 2020, proceedings, part V*, vol. 12350 of *Lecture Notes in Computer Science*, 491–507. Springer.

Kolter, Zico, and Aleksander Madry. 2018. Adversarial Robustness - Theory and Practice. `https://adversarial-ml-tutorial.org/`.

Kornblith, Simon, Mohammad Norouzi, Honglak Lee, and Geoffrey Hinton. 2019. Similarity of neural network representations revisited. In *International conference on machine learning*, 3519–3529. PMLR.

Kotar, Klemen, Gabriel Ilharco, Ludwig Schmidt, Kiana Ehsani, and Roozbeh Mottaghi. 2021. Contrasting contrastive self-supervised representation learning pipelines. In *2021 IEEE/CVF international conference on computer vision, ICCV 2021, montreal, qc, canada, october 10-17, 2021*, 9929–9939. IEEE.

Krizhevsky, Alex, Geoffrey Hinton, et al. 2009. Learning multiple layers of features from tiny images.

Krizhevsky, Alex, Ilya Sutskever, and Geoffrey E. Hinton. 2012. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems 25: 26th annual conference on neural information processing systems 2012. proceedings of a meeting held december 3-6, 2012, lake tahoe, nevada, united states*, ed. Peter L. Bartlett, Fernando C. N. Pereira, Christopher J. C. Burges, Léon Bottou, and Kilian Q. Weinberger, 1106–1114.

Kuhn, Lorenz, Yarin Gal, and Sebastian Farquhar. 2023. Semantic uncertainty: Linguistic invariances for uncertainty estimation in natural language generation. In *The eleventh international conference on learning representations*.

Laidlaw, Cassidy, and Soheil Feizi. 2019a. Functional adversarial attacks. In *Neurips*, 10408–10418.

———. 2019b. Playing it safe: Adversarial robustness with an abstain option. *CoRR* abs/1911.11253. 1911.11253.

Laidlaw, Cassidy, Sahil Singla, and Soheil Feizi. 2021. Perceptual adversarial robustness: Defense against unseen threat models. In *9th international conference on learning representations, ICLR 2021, virtual event, austria, may 3-7, 2021*. OpenReview.net.

Lakshminarayanan, Balaji, Alexander Pritzel, and Charles Blundell. 2017. Simple and scalable predictive uncertainty estimation using deep ensembles. In *Neurips*, 6402–6413.

LeCun, Yann. 1998. The MNIST database of handwritten digits.

LeCun, Yann, Bernhard E. Boser, John S. Denker, Donnie Henderson, Richard E. Howard, Wayne E. Hubbard, and Lawrence D. Jackel. 1989. Handwritten digit recognition with a back-propagation network. In *Advances in neural information processing systems 2,* [*NIPS conference, denver, colorado, usa, november 27-30, 1989*], ed. David S. Touretzky, 396–404. Morgan Kaufmann.

Lee, Dong-Hyun, et al. 2013. Pseudo-label: The simple and efficient semi-supervised learning method for deep neural networks. In *Workshop on challenges in representation learning, icml*, vol. 3, 896.

Lee, Doyup, Sungwoong Kim, Ildoo Kim, Yeongjae Cheon, Minsu Cho, and Wook-Shin Han. 2022. Contrastive regularization for semi-supervised learning. In *Proceedings of the ieee/cvf conference on computer vision and pattern recognition*, 3911–3920.

Lee, Jason D, Qi Lei, Nikunj Saunshi, and Jiacheng Zhuo. 2021. Predicting what you already know helps: Provable self-supervised learning. *Advances in Neural Information Processing Systems* 34:309–323.

Lee, Kimin, Honglak Lee, Kibok Lee, and Jinwoo Shin. 2018a. Training confidence-calibrated classifiers for detecting out-of-distribution samples. *ICLR*.

Lee, Kimin, Kibok Lee, Honglak Lee, and Jinwoo Shin. 2018b. A simple unified framework for detecting out-of-distribution samples and adversarial attacks. In *Neurips*, 7167–7177.

Lester, Brian, Rami Al-Rfou, and Noah Constant. 2021. The power of scale for parameter-efficient prompt tuning. In *Proceedings of the 2021 conference on empirical methods in natural language processing*, 3045–3059.

Liang, Shiyu, Yixuan Li, and Rayadurgam Srikant. 2018. Enhancing the reliability of out-of-distribution image detection in neural networks. In *Iclr*.

Lin, Chin-Yew, and Franz Josef Och. 2004. Automatic evaluation of machine translation quality using longest common subsequence and skip-bigram statistics. In *Proceedings of the 42nd annual meeting of the association for computational linguistics (acl-04)*, 605–612.

Lin, Ziqian, Sreya Dutta, and Yixuan Li. 2021. Mood: Multi-level out-of-distribution detection. *CVPR*.

Liu, Hong, Jeff Z HaoChen, Adrien Gaidon, and Tengyu Ma. 2021a. Self-supervised learning is more robust to dataset imbalance. In *Neurips 2021 workshop on distribution shifts: Connecting methods and applications*.

Liu, Weitang, Xiaoyun Wang, John Owens, and Yixuan Li. 2020. Energy-based out-of-distribution detection. *NeurIPS*.

Liu, Xiao, Kaixuan Ji, Yicheng Fu, Weng Lam Tam, Zhengxiao Du, Zhilin Yang, and Jie Tang. 2021b. P-tuning v2: Prompt tuning can be comparable to fine-tuning universally across scales and tasks. *arXiv preprint arXiv:2110.07602*.

Liu, Xiao, Yanan Zheng, Zhengxiao Du, Ming Ding, Yujie Qian, Zhilin Yang, and Jie Tang. 2021c. Gpt understands, too. *arXiv preprint arXiv:2103.10385*.

Liu, Xiaofeng, Chaehwa Yoo, Fangxu Xing, Hyejin Oh, Georges El Fakhri, Je-Won Kang, Jonghye Woo, et al. 2022. Deep unsupervised domain adaptation: A review of recent advances and perspectives. *APSIPA Transactions on Signal and Information Processing* 11(1).

Liu, Yanpei, Xinyun Chen, Chang Liu, and Dawn Song. 2016. Delving into transferable adversarial examples and black-box attacks. *CoRR* abs/1611.02770. 1611.02770.

Liu, Yinhan, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

Lo Piano, Samuele. 2020. Ethical principles in machine learning and artificial intelligence: cases from the field and possible ways forward. *Humanities and Social Sciences Communications* 7.

Long, Mingsheng, Zhangjie Cao, Jianmin Wang, and Michael I Jordan. 2018. Conditional adversarial domain adaptation. *Advances in neural information processing systems* 31.

Lorraine, Jonathan, and David Duvenaud. 2018. Stochastic hyperparameter optimization through hypernetworks. *CoRR* abs/1802.09419.

Luenberger, David G. 1997. *Optimization by vector space methods*. 1st ed. New York, NY, USA: John Wiley & Sons, Inc.

Ma, Kaili, Haochen Yang, Han Yang, Tatiana Jin, Pengfei Chen, Yongqiang Chen, Barakeel Fanseu Kamhoua, and James Cheng. 2021. Improving graph representation learning by contrastive regularization. *arXiv preprint arXiv:2101.11525*.

Van der Maaten, Laurens, and Geoffrey Hinton. 2008. Visualizing data using t-sne. *Journal of machine learning research* 9(11).

Madry, Aleksander, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. 2018. Towards deep learning models resistant to adversarial attacks.

In *6th international conference on learning representations, ICLR 2018, vancouver, bc, canada, april 30 - may 3, 2018, conference track proceedings*. OpenReview.net.

Malinin, Andrey, and Mark Gales. 2018. Predictive uncertainty estimation via prior networks. In *Neurips*, 7047–7058.

Markoff, John. 2016. For now, self-driving cars still need humans. `https://www.nytimes.com/2016/01/18/technology/driverless-cars-limits-include-human-nature.html`. The New York Times.

McCallum, Andrew, Kamal Nigam, et al. 1998. Employing em and pool-based active learning for text classification. In *Icml*, vol. 98, 350–358. Citeseer.

Meinke, Alexander, and Matthias Hein. 2020. Towards neural networks that provably know when they don't know. In *8th international conference on learning representations, ICLR 2020, addis ababa, ethiopia, april 26-30, 2020*. OpenReview.net.

Moghimi, Mohammad, Serge J Belongie, Mohammad J Saberian, Jian Yang, Nuno Vasconcelos, and Li-Jia Li. 2016. Boosted convolutional neural networks. In *Bmvc*, vol. 5, 6.

Mohajerin Esfahani, Peyman, and Daniel Kuhn. 2015. Data-driven distributionally robust optimization using the wasserstein metric: performance guarantees and tractable reformulations. *arXiv preprint arXiv:1505.05116*.

Mohseni, Sina, Mandar Pitale, J. B. S. Yadawa, and Zhangyang Wang. 2020. Self-supervised learning for generalizable out-of-distribution detection. In *The thirty-fourth AAAI conference on artificial intelligence, AAAI 2020, the thirty-second innovative applications of artificial intelligence conference, IAAI 2020, the tenth AAAI symposium on educational advances in artificial intelligence, EAAI 2020, new york, ny, usa, february 7-12, 2020*, 5216–5223. AAAI Press.

Mozannar, Hussein, and David Sontag. 2020. Consistent estimators for learning to defer to an expert. In *International conference on machine learning*, 7076–7087. PMLR.

Najafi, Amir, Shin-ichi Maeda, Masanori Koyama, and Takeru Miyato. 2019. Robustness to adversarial perturbations in learning from incomplete data. In *Neurips*, 5541–5551.

Netzer, Yuval, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Y Ng. 2011. Reading digits in natural images with unsupervised feature learning.

Newell, Alejandro, and Jia Deng. 2020. How useful is self-supervised pretraining for visual tasks? In *2020 IEEE/CVF conference on computer vision and pattern recognition, CVPR 2020, seattle, wa, usa, june 13-19, 2020*, 7343–7352. Computer Vision Foundation / IEEE.

Ni, Jianmo, Jiacheng Li, and Julian McAuley. 2019. Justifying recommendations using distantly-labeled reviews and fine-grained aspects. In *Proceedings of the 2019 conference on empirical methods in natural language processing and the 9th international joint conference on natural language processing (emnlp-ijcnlp)*, 188–197.

Nilsback, M-E, and Andrew Zisserman. 2006. A visual vocabulary for flower classification. In *2006 ieee computer society conference on computer vision and pattern recognition (cvpr'06)*, vol. 2, 1447–1454. IEEE.

Olshausen, B., and D. Field. 1997. Sparse coding with an overcomplete basis set: A strategy employed by v1? *Vision Research* 37:3311–3325.

van den Oord, Aäron, Yazhe Li, and Oriol Vinyals. 2018. Representation learning with contrastive predictive coding. *CoRR* abs/1807.03748. `1807.03748`.

OpenAI, R. 2023. Gpt-4 technical report. *arXiv*.

Ovadia, Yaniv, Emily Fertig, Jie Ren, Zachary Nado, David Sculley, Sebastian Nowozin, Joshua Dillon, Balaji Lakshminarayanan, and Jasper Snoek. 2019. Can you trust your model's uncertainty? evaluating predictive uncertainty under dataset shift. In *Advances in neural information processing systems*, 13991–14002.

Pang, Tianyu, Huishuai Zhang, Di He, Yinpeng Dong, Hang Su, Wei Chen, Jun Zhu, and Tie-Yan Liu. 2022. Two coupled rejection metrics can tell adversarial examples apart. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 15223–15233.

Papadopoulos, Aristotelis-Angelos, Mohammad Reza Rajati, Nazim Shaikh, and Jiamian Wang. 2021. Outlier exposure with confidence control for out-of-distribution detection. *Neurocomputing* 441:138–150.

Papernot, Nicolas, Patrick McDaniel, Somesh Jha, Matt Fredrikson, Z Berkay Celik, and Ananthram Swami. 2016. The limitations of deep learning in adversarial settings. In *2016 ieee european symposium on security and privacy (euros&p)*, 372–387. IEEE.

Pearson, Karl. 1901. LIII. On lines and planes of closest fit to systems of points in space. *The London, Edinburgh, and Dublin philosophical magazine and journal of science* 2(11):559–572.

Pedregosa, F., 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. 2011. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research* 12:2825–2830.

Peng, Xingchao, Qinxun Bai, Xide Xia, Zijun Huang, Kate Saenko, and Bo Wang. 2019. Moment matching for multi-source domain adaptation. In *Proceedings of the ieee international conference on computer vision*, 1406–1415.

Peters, Matthew E., Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of the 2018 conference of the north american chapter of the association for computational linguistics: Human language technologies, NAACL-HLT 2018, new orleans, louisiana, usa, june 1-6, 2018, volume 1 (long papers)*, ed. Marilyn A. Walker, Heng Ji, and Amanda Stent, 2227–2237. Association for Computational Linguistics.

Platanios, Emmanouil A., Hoifung Poon, Tom M. Mitchell, and Eric Horvitz. 2017. Estimating accuracy from unlabeled data: A probabilistic logic approach. In *Advances in neural information processing systems 30: Annual conference on neural information processing systems 2017, december 4-9, 2017, long beach, ca, USA*, ed. Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett, 4361–4370.

Platanios, Emmanouil Antonios, Avrim Blum, and Tom M. Mitchell. 2014. Estimating accuracy from unlabeled data. In *Proceedings of the thirtieth conference on uncertainty in artificial intelligence, UAI 2014, quebec city, quebec, canada, july 23-27, 2014*, ed. Nevin L. Zhang and Jin Tian, 682–691. AUAI Press.

Poesia, Gabriel, Oleksandr Polozov, Vu Le, Ashish Tiwari, Gustavo Soares, Christopher Meek, and Sumit Gulwani. 2022. Synchromesh: Reliable code generation from pre-trained language models. *arXiv preprint arXiv:2201.11227*.

Prabhu, Viraj, Arjun Chandrasekaran, Kate Saenko, and Judy Hoffman. 2021. Active domain adaptation via clustering uncertainty-weighted embeddings. In *Proceedings of the ieee/cvf international conference on computer vision*, 8505–8514.

Quiñonero-Candela, Joaquin, Masashi Sugiyama, Anton Schwaighofer, and Neil D Lawrence. 2008. *Dataset shift in machine learning*. Mit Press.

Rabanser, Stephan, Anvith Thudi, Kimia Hamidieh, Adam Dziedzic, and Nicolas Papernot. 2022. Selective classification via neural network training dynamics. *arXiv preprint arXiv:2205.13532*.

Radford, Alec, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. 2021. Learning transferable visual models from natural language supervision. In *Proceedings of the 38th international conference on machine learning, ICML 2021, 18-24 july 2021, virtual event*, ed. Marina Meila and Tong Zhang, vol. 139 of *Proceedings of Machine Learning Research*, 8748–8763. PMLR.

Radford, Alec, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. 2019. Language models are unsupervised multitask learners. *OpenAI blog* 1(8):9.

Rajpurkar, Pranav, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. Squad: 100, 000+ questions for machine comprehension of text. In *Proceedings of the 2016 conference on empirical methods in natural language processing, EMNLP 2016, austin, texas, usa, november 1-4, 2016*, ed. Jian Su, Xavier Carreras, and Kevin Duh, 2383–2392. The Association for Computational Linguistics.

Ramesh, Aditya, Prafulla Dhariwal, Alex Nichol, Casey Chu, and Mark Chen. 2022. Hierarchical text-conditional image generation with clip latents. *arXiv preprint arXiv:2204.06125*.

Reddy, Siva, Danqi Chen, and Christopher D Manning. 2019. Coqa: A conversational question answering challenge. *Transactions of the Association for Computational Linguistics* 7:249–266.

Ren, Jie, Jiaming Luo, Yao Zhao, Kundan Krishna, Mohammad Saleh, Balaji Lakshminarayanan, and Peter J Liu. 2022. Out-of-distribution detection and selective generation for conditional language models. *arXiv preprint arXiv:2209.15558*.

Ren, Shaoqing, Kaiming He, Ross B. Girshick, and Jian Sun. 2015. Faster R-CNN: towards real-time object detection with region proposal networks. In *Advances in neural information processing systems 28: Annual conference on neural information processing systems 2015, december 7-12, 2015, montreal, quebec, canada*, ed. Corinna Cortes, Neil D. Lawrence, Daniel D. Lee, Masashi Sugiyama, and Roman Garnett, 91–99.

Rockafellar, R Tyrrell, and Roger J-B Wets. 2009. *Variational analysis*, vol. 317. Springer Science & Business Media.

Ross, Andrew Slavin, and Finale Doshi-Velez. 2018. Improving the adversarial robustness and interpretability of deep neural networks by regularizing their input

gradients. In *Proceedings of the thirty-second AAAI conference on artificial intelligence*, (*aaai-18*), 1660–1669.

Saenko, Kate, Brian Kulis, Mario Fritz, and Trevor Darrell. 2010. Adapting visual category models to new domains. In *European conference on computer vision*, 213–226. Springer.

Sagawa, Shiori, Pang Wei Koh, Tony Lee, Irena Gao, Sang Michael Xie, Kendrick Shen, Ananya Kumar, Weihua Hu, Michihiro Yasunaga, Henrik Marklund, et al. 2021. Extending the wilds benchmark for unsupervised adaptation. *arXiv preprint arXiv:2112.05090*.

Saito, Kuniaki, Donghyun Kim, Stan Sclaroff, Trevor Darrell, and Kate Saenko. 2019. Semi-supervised domain adaptation via minimax entropy. In *Proceedings of the ieee/cvf international conference on computer vision*, 8050–8058.

Salehi, Mohammadreza, Hossein Mirzaei, Dan Hendrycks, Yixuan Li, Mohammad Hossein Rohban, and Mohammad Sabokrou. 2021. A unified survey on anomaly, novelty, open-set, and out-of-distribution detection: Solutions and future challenges. *arXiv preprint arXiv:2110.14051*.

Saunshi, Nikunj, Jordan Ash, Surbhi Goel, Dipendra Misra, Cyril Zhang, Sanjeev Arora, Sham Kakade, and Akshay Krishnamurthy. 2022. Understanding contrastive learning requires incorporating inductive biases. *arXiv preprint arXiv:2202.14037*.

Schelter, Sebastian, Tammo Rukat, and Felix Biessmann. 2020. Learning to validate the predictions of black box classifiers on unseen data. In *Proceedings of the 2020 acm sigmod international conference on management of data*, 1289–1299.

Schmidt, Ludwig, Shibani Santurkar, Dimitris Tsipras, Kunal Talwar, and Aleksander Madry. 2018. Adversarially robust generalization requires more data. In *Advances in neural information processing systems*, 5014–5026.

Sehwag, Vikash, Arjun Nitin Bhagoji, Liwei Song, Chawin Sitawarin, Daniel Cullina, Mung Chiang, and Prateek Mittal. 2019. Analyzing the robustness of open-world machine learning. In *Proceedings of the 12th acm workshop on artificial intelligence and security*, 105–116.

Sehwag, Vikash, Shiqi Wang, Prateek Mittal, and Suman Jana. 2020. HYDRA: pruning adversarially robust neural networks. In *Advances in neural information processing systems 33: Annual conference on neural information processing systems 2020, neurips 2020, december 6-12, 2020, virtual*, ed. Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin.

Sener, Ozan, and Silvio Savarese. 2017. Active learning for convolutional neural networks: A core-set approach. *arXiv preprint arXiv:1708.00489*.

Settles, Burr. 2009. Active learning literature survey.

Sharma, Prinkle, David Austin, and Hong Liu. 2019. Attacks on machine learning: Adversarial examples in connected and autonomous vehicles. In *2019 ieee international symposium on technologies for homeland security (hst)*, 1–7. IEEE.

Sheikholeslami, Fatemeh, Ali Lotfi, and J. Zico Kolter. 2021. Provably robust classification of adversarial examples with detection. In *9th international conference on learning representations (ICLR)*. OpenReview.net.

Shen, Kendrick, Robbie M Jones, Ananya Kumar, Sang Michael Xie, Jeff Z HaoChen, Tengyu Ma, and Percy Liang. 2022. Connect, not collapse: Explaining contrastive learning for unsupervised domain adaptation. In *International conference on machine learning*, 19847–19878. PMLR.

Shi, Zhenmei, Jiefeng Chen, Kunyang Li, Jayaram Raghuram, Xi Wu, Yingyu Liang, and Somesh Jha. 2022a. The trade-off between universality and label efficiency of representations from contrastive learning. In *The eleventh international conference on learning representations*.

Shi, Zhenmei, Junyi Wei, and Yingyu Liang. 2022b. A theoretical analysis on feature learning in neural networks: Emergence from inputs and advantage over fixed features. In *International conference on learning representations*.

Shrivastava, Abhinav, Abhinav Gupta, and Ross Girshick. 2016. Training region-based object detectors with online hard example mining. In *Cvpr*, 761–769.

Si, Chenglei, Zhe Gan, Zhengyuan Yang, Shuohang Wang, Jianfeng Wang, Jordan Boyd-Graber, and Lijuan Wang. 2022. Prompting gpt-3 to be reliable. *arXiv preprint arXiv:2210.09150*.

Silver, David, Aja Huang, Chris J. Maddison, Arthur Guez, Laurent Sifre, George van den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Vedavyas Panneershelvam, Marc Lanctot, Sander Dieleman, Dominik Grewe, John Nham, Nal Kalchbrenner, Ilya Sutskever, Timothy P. Lillicrap, Madeleine Leach, Koray Kavukcuoglu, Thore Graepel, and Demis Hassabis. 2016. Mastering the game of go with deep neural networks and tree search. *Nat.* 529(7587):484–489.

Silver, David, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel, Timothy P. Lillicrap, Karen Simonyan, and Demis Hassabis. 2017. Mastering chess and shogi by self-play with a general reinforcement learning algorithm. *CoRR* abs/1712.01815. 1712.01815.

Simo-Serra, Edgar, Eduard Trulls, Luis Ferraz, Iasonas Kokkinos, Pascal Fua, and Francesc Moreno-Noguer. 2015. Discriminative learning of deep convolutional feature point descriptors. In *Iccv*, 118–126.

Singhal, Karan, Shekoofeh Azizi, Tao Tu, S Sara Mahdavi, Jason Wei, Hyung Won Chung, Nathan Scales, Ajay Tanwani, Heather Cole-Lewis, Stephen Pfohl, et al. 2022. Large language models encode clinical knowledge. *arXiv preprint arXiv:2212.13138*.

Singhal, Karan, Tao Tu, Juraj Gottweis, Rory Sayres, Ellery Wulczyn, Le Hou, Kevin Clark, Stephen Pfohl, Heather Cole-Lewis, Darlene Neal, et al. 2023. Towards

expert-level medical question answering with large language models. *arXiv preprint arXiv:2305.09617*.

Sinha, Aman, Hongseok Namkoong, and John C. Duchi. 2018. Certifying some distributional robustness with principled adversarial training. In *6th international conference on learning representations, ICLR 2018, vancouver, bc, canada, april 30 - may 3, 2018, conference track proceedings*. OpenReview.net.

Sinha, Samarth, Sayna Ebrahimi, and Trevor Darrell. 2019. Variational adversarial active learning. In *Proceedings of the ieee/cvf international conference on computer vision*, 5972–5981.

Smirnov, Evgeny, Aleksandr Melnikov, Andrei Oleinik, Elizaveta Ivanova, Ilya Kalinovskiy, and Eugene Luckyanets. 2018. Hard example mining with auxiliary embeddings. In *Cvpr workshops*, 37–46.

Sohn, Kihyuk, David Berthelot, Nicholas Carlini, Zizhao Zhang, Han Zhang, Colin A Raffel, Ekin Dogus Cubuk, Alexey Kurakin, and Chun-Liang Li. 2020. Fixmatch: Simplifying semi-supervised learning with consistency and confidence. *Advances in neural information processing systems* 33:596–608.

Šrndic, Nedim, and Pavel Laskov. 2013. Detection of malicious pdf files based on hierarchical document structure. In *Proceedings of the 20th annual network & distributed system security symposium*, 1–16. Citeseer.

Stallkamp, Johannes, Marc Schlipsing, Jan Salmen, and Christian Igel. 2012. Man vs. computer: Benchmarking machine learning algorithms for traffic sign recognition. *Neural Networks* 32:323–332.

Steinhardt, Jacob, and Percy S Liang. 2016. Unsupervised risk estimation using only conditional independence structure. In *Advances in neural information processing systems*, 3657–3665.

Stutz, David, Matthias Hein, and Bernt Schiele. 2020. Confidence-calibrated adversarial training: Generalizing to unseen attacks. In *Proceedings of the 37th*

international conference on machine learning (ICML), vol. 119 of *Proceedings of Machine Learning Research*, 9155–9166. PMLR.

Su, Jong-Chyi, Yi-Hsuan Tsai, Kihyuk Sohn, Buyu Liu, Subhransu Maji, and Manmohan Chandraker. 2020. Active adversarial domain adaptation. In *Proceedings of the ieee/cvf winter conference on applications of computer vision*, 739–748.

Subramanya, Akshayvarun, Suraj Srinivas, and R Venkatesh Babu. 2017. Confidence estimation in deep neural networks via density modelling. *arXiv preprint arXiv:1707.07013*.

Suh, Yumin, Bohyung Han, Wonsik Kim, and Kyoung Mu Lee. 2019. Stochastic class-based hard example mining for deep metric learning. In *Cvpr*, 7251–7259.

Sundararajan, Mukund, Ankur Taly, and Qiqi Yan. 2017. Axiomatic attribution for deep networks. In *Proceedings of the 34th international conference on machine learning*, 3319–3328.

Sung, Kah Kay. 1995. Learning and example selection for object and pattern detection. Ph.D. thesis, Massachusetts Institute of Technology, Cambridge, MA, USA.

Sung, Yi-Lin, Jaemin Cho, and Mohit Bansal. 2022. Vl-adapter: Parameter-efficient transfer learning for vision-and-language tasks. In *Proceedings of the ieee/cvf conference on computer vision and pattern recognition*, 5227–5237.

Szegedy, Christian, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian J. Goodfellow, and Rob Fergus. 2014. Intriguing properties of neural networks. In *2nd international conference on learning representations, ICLR 2014, banff, ab, canada, april 14-16, 2014, conference track proceedings*, ed. Yoshua Bengio and Yann LeCun.

Tang, Liyan, Zhaoyi Sun, Betina Idnay, Jordan G Nestor, Ali Soroush, Pierre A Elias, Ziyang Xu, Ying Ding, Greg Durrett, Justin Rousseau, et al. 2023. Evaluating large language models on medical evidence summarization. *medRxiv* 2023–04.

Tax, David M. J., and Robert P. W. Duin. 2008. Growing a multi-class classifier with a reject option. *Pattern Recognition Letters* 29(10):1565–1570.

Tian, Yuandong. 2022. Deep contrastive learning is provably (almost) principal component analysis. *arXiv preprint arXiv:2201.12680*.

Torralba, Antonio, and Alexei A Efros. 2011. Unbiased look at dataset bias. In *Cvpr 2011*, 1521–1528. IEEE.

Torralba, Antonio, Rob Fergus, and William T Freeman. 2008. 80 million tiny images: A large data set for nonparametric object and scene recognition. *IEEE transactions on pattern analysis and machine intelligence* 30(11):1958–1970.

Tosh, Christopher, Akshay Krishnamurthy, and Daniel Hsu. 2021. Contrastive learning, multi-view redundancy, and linear models. In *Algorithmic learning theory*, 1179–1206. PMLR.

Tramèr, Florian. 2021. Detecting adversarial examples is (nearly) as hard as classifying them. *CoRR* abs/2107.11630. `2107.11630`.

Tramèr, Florian, Nicholas Carlini, Wieland Brendel, and Aleksander Madry. 2020. On adaptive attacks to adversarial example defenses. In *Advances in neural information processing systems 33: Annual conference on neural information processing systems*.

Tramèr, Florian, Nicolas Papernot, Ian Goodfellow, Dan Boneh, and Patrick McDaniel. 2017. The space of transferable adversarial examples. *arXiv*.

Tsai, Yao-Hung Hubert, Yue Wu, Ruslan Salakhutdinov, and Louis-Philippe Morency. 2020. Self-supervised learning from a multi-view perspective. In *International conference on learning representations*.

Tsipras, Dimitris, Shibani Santurkar, Logan Engstrom, Alexander Turner, and Aleksander Madry. 2019. Robustness may be at odds with accuracy. In *International conference on learning representations*.

Uesato, Jonathan, Jean-Baptiste Alayrac, Po-Sen Huang, Robert Stanforth, Alhussein Fawzi, and Pushmeet Kohli. 2019. Are labels required for improving adversarial robustness? *NeurIPS*.

Van Gansbeke, Wouter, Simon Vandenhende, Stamatios Georgoulis, and Luc V Gool. 2021. Revisiting contrastive methods for unsupervised learning of visual representations. *Advances in Neural Information Processing Systems* 34:16238–16250.

Vapnik, Vladimir. 1998. *Statistical learning theory*. Wiley.

Varshney, Neeraj, Swaroop Mishra, and Chitta Baral. 2022. Investigating selective prediction approaches across several tasks in iid, ood, and adversarial settings. In *Findings of the association for computational linguistics: Acl 2022*, 1995–2002.

Vaswani, Ashish, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in neural information processing systems 30: Annual conference on neural information processing systems 2017, december 4-9, 2017, long beach, ca, USA*, ed. Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett, 5998–6008.

Wang, Dequan, An Ju, Evan Shelhamer, David Wagner, and Trevor Darrell. 2021a. Fighting gradients with gradients: Dynamic defenses against adversarial attacks. *arXiv preprint arXiv:2105.08714*.

Wang, Feng, Guoyizhe Wei, Qiao Liu, Jinxiang Ou, Hairong Lv, et al. 2021b. Boost neural networks by checkpoints. *Advances in Neural Information Processing Systems* 34:19719–19729.

Wang, Tongzhou, and Phillip Isola. 2020. Understanding contrastive representation learning through alignment and uniformity on the hypersphere. In *International conference on machine learning*, 9929–9939. PMLR.

Wang, Xiaolong, and Abhinav Gupta. 2015. Unsupervised learning of visual representations using videos. In *Iccv*, 2794–2802.

Wang, Xuezhi, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, and Denny Zhou. 2022. Self-consistency improves chain of thought reasoning in language models. *arXiv preprint arXiv:2203.11171*.

Wang, Yisen, Difan Zou, Jinfeng Yi, James Bailey, Xingjun Ma, and Quanquan Gu. 2020. Improving adversarial robustness requires revisiting misclassified examples. In *8th international conference on learning representations, ICLR 2020, addis ababa, ethiopia, april 26-30, 2020*. OpenReview.net.

Wei, Colin, Kendrick Shen, Yining Chen, and Tengyu Ma. 2020. Theoretical analysis of self-training with deep networks on unlabeled data. In *International conference on learning representations*.

Wen, Zixin, and Yuanzhi Li. 2021. Toward understanding the feature learning process of self-supervised contrastive learning. In *International conference on machine learning*, 11112–11122. PMLR.

Williams, Adina, Nikita Nangia, and Samuel R. Bowman. 2018. A broad-coverage challenge corpus for sentence understanding through inference. In *Proceedings of the 2018 conference of the north american chapter of the association for computational linguistics: Human language technologies, NAACL-HLT 2018, new orleans, louisiana, usa, june 1-6, 2018, volume 1 (long papers)*, ed. Marilyn A. Walker, Heng Ji, and Amanda Stent, 1112–1122. Association for Computational Linguistics.

Wilson, Garrett, and Diane J Cook. 2020. A survey of unsupervised deep domain adaptation. *ACM Transactions on Intelligent Systems and Technology (TIST)* 11(5): 1–46.

Wong, Eric, and J. Zico Kolter. 2018. Provable defenses against adversarial examples via the convex outer adversarial polytope. In *Proceedings of the 35th international conference on machine learning, ICML 2018*, 5283–5292.

Wong, Eric, Leslie Rice, and J. Zico Kolter. 2020. Fast is better than free: Revisiting adversarial training. In *8th international conference on learning representations, ICLR 2020, addis ababa, ethiopia, april 26-30, 2020*. OpenReview.net.

Wu, Chao-Yuan, R Manmatha, Alexander J Smola, and Philipp Krahenbuhl. 2017. Sampling matters in deep embedding learning. In *Iccv*, 2840–2848.

Wu, Dongxian, Shu-Tao Xia, and Yisen Wang. 2020a. Adversarial weight perturbation helps robust generalization. In *Advances in neural information processing systems 33: Annual conference on neural information processing systems 2020, neurips 2020, december 6-12, 2020, virtual*, ed. Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin.

Wu, Yi-Hsuan, Chia-Hung Yuan, and Shan-Hung Wu. 2020b. Adversarial robustness via runtime masking and cleansing. In *Proceedings of the 37th international conference on machine learning, ICML 2020, 13-18 july 2020, virtual event*, vol. 119 of *Proceedings of Machine Learning Research*, 10399–10409. PMLR.

Xiao, Han, Kashif Rasul, and Roland Vollgraf. 2017. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747*.

Xin, Ji, Raphael Tang, Yaoliang Yu, and Jimmy Lin. 2021. The art of abstention: Selective prediction and error regularization for natural language processing. In *Proceedings of the 59th annual meeting of the association for computational linguistics and the 11th international joint conference on natural language processing (volume 1: Long papers)*, 1040–1051.

Xu, Pingmei, Krista A Ehinger, Yinda Zhang, Adam Finkelstein, Sanjeev R Kulkarni, and Jianxiong Xiao. 2015. Turkergaze: Crowdsourcing saliency with webcam based eye tracking. *arXiv preprint arXiv:1504.06755*.

Yang, Wanqian. 2020. Making decisions under high stakes: Trustworthy and expressive bayesian deep learning. Ph.D. thesis.

Yang, Xingyi, Xuehai He, Yuxiao Liang, Yue Yang, Shanghang Zhang, and Pengtao Xie. 2020. Transfer learning or self-supervised learning? A tale of two pretraining paradigms. *CoRR* abs/2007.04234. 2007.04234.

Yao, Huaxiu, Caroline Choi, Bochuan Cao, Yoonho Lee, Pang Wei Koh, and Chelsea Finn. 2022. Wild-time: A benchmark of in-the-wild distribution shift over time. *arXiv preprint arXiv:2211.14238*.

Yarowsky, David. 1995. Unsupervised word sense disambiguation rivaling supervised methods. In *33rd annual meeting of the association for computational linguistics*, 189–196.

Yin, Xuwang, Soheil Kolouri, and Gustavo K. Rohde. 2020. GAT: generative adversarial training for adversarial example detection and robust classification. In *8th international conference on learning representations (ICLR)*. OpenReview.net.

Yin, Zhangyue, Qiushi Sun, Qipeng Guo, Jiawen Wu, Xipeng Qiu, and Xuanjing Huang. 2023. Do large language models know what they don't know? *arXiv preprint arXiv:2305.18153*.

Yu, Fisher, Ari Seff, Yinda Zhang, Shuran Song, Thomas Funkhouser, and Jianxiong Xiao. 2015. Lsun: Construction of a large-scale image dataset using deep learning with humans in the loop. *arXiv preprint arXiv:1506.03365*.

Yuan, Michelle, Hsuan-Tien Lin, and Jordan Boyd-Graber. 2020. Cold-start active learning through self-supervised language modeling. *arXiv preprint arXiv:2010.09535*.

Yuan, Yuhui, Kuiyuan Yang, and Chao Zhang. 2017. Hard-aware deeply cascaded embedding. In *Iccv*, 814–823.

Zbontar, Jure, Li Jing, Ishan Misra, Yann LeCun, and Stéphane Deny. 2021. Barlow twins: Self-supervised learning via redundancy reduction. In *Proceedings of the 38th international conference on machine learning, ICML 2021, 18-24 july 2021, virtual event*, vol. 139 of *Proceedings of Machine Learning Research*, 12310–12320. PMLR.

Zhai, Runtian, Tianle Cai, Di He, Chen Dan, Kun He, John Hopcroft, and Liwei Wang. 2019. Adversarially robust generalization just requires more unlabeled data. *arXiv preprint arXiv:1906.00555*.

Zhang, Hongyang, Yaodong Yu, Jiantao Jiao, Eric P. Xing, Laurent El Ghaoui, and Michael I. Jordan. 2019. Theoretically principled trade-off between robustness and accuracy. In *Proceedings of the 36th international conference on machine learning (ICML)*, vol. 97 of *Proceedings of Machine Learning Research*, 7472–7482. PMLR.

Zhang, Shun, Zhenfang Chen, Yikang Shen, Mingyu Ding, Joshua B Tenenbaum, and Chuang Gan. 2023a. Planning with large language models for code generation. *arXiv preprint arXiv:2303.05510*.

Zhang, Susan, Stephen Roller, Naman Goyal, Mikel Artetxe, Moya Chen, Shuohui Chen, Christopher Dewan, Mona Diab, Xian Li, Xi Victoria Lin, et al. 2022. Opt: Open pre-trained transformer language models. *arXiv preprint arXiv:2205.01068*.

Zhang, Tianyi, Faisal Ladhak, Esin Durmus, Percy Liang, Kathleen McKeown, and Tatsunori B Hashimoto. 2023b. Benchmarking large language models for news summarization. *arXiv preprint arXiv:2301.13848*.

Zhao, Eric, Anqi Liu, Animashree Anandkumar, and Yisong Yue. 2021. Active learning under label shift. In *International conference on artificial intelligence and statistics*, 3412–3420. PMLR.

Zhou, Bolei, Agata Lapedriza, Aditya Khosla, Aude Oliva, and Antonio Torralba. 2017. Places: A 10 million image database for scene recognition. *IEEE transactions on pattern analysis and machine intelligence* 40(6):1452–1464.

Zhu, Xiatian, Shaogang Gong, et al. 2018. Knowledge distillation by on-the-fly native ensemble. *Advances in neural information processing systems* 31.

Zimmermann, Roland S, Yash Sharma, Steffen Schneider, Matthias Bethge, and Wieland Brendel. 2021. Contrastive learning inverts the data generating process. In *International conference on machine learning*, 12979–12990. PMLR.