Exploiting Structure for Designing Clinical Trials: Testing, Learning and Inference Algorithms

by

Vamsi K. Ithapu

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

Doctor of Philosophy

(Computer Sciences)

at the

UNIVERSITY OF WISCONSIN–MADISON

2018

Date of final oral examination: 01/24/2018

The dissertation is approved by the following members of the Final Oral Committee:
Vikas Singh, Associate Professor, Biostatistics & Medical Informatics, Computer Sciences
Sterling C. Johnson, Professor, Medicine
Xiaojin Zhu, Professor, Computer Sciences
Charles R. Dyer, Professor, Computer Sciences, Biostatistics & Medical Informatics
Mohit Gupta, Assistant Professor, Computer Sciences

To the many giants,
on whose shoulders, I humbly stand

Acknowledgments

The six years that I spent in the city of four lakes have been quite memorable. While I did pursue my decade-old dream of earning a doctoral degree, these years are indelible primarily because of the delightful and exuberant people I met, who inadvertently shaped my professional as well as personal life.

My desire to pursue a Ph.D. degree began over 15 years ago when I barely knew what research meant! During those high school years, my uncle, Mohan Perugupalli, had a strong influence on me. An Electrical Engineer working for Indian government research, his stories of doing "scientific stuff" – my childish interpretation of science/engineering at the time, and the few times I was fortunate to visit his workplace, have influenced me a lot. They stayed with me all through my childhood. On multiple occasions, he told me that excelling in Mathematics and Physics is where I should begin. While in higher secondary, because of friends and teachers, I stumbled upon an abbreviation IIT – Indian Institute of Technology – an elite group of schools for pursuing bachelors in India. With extensive support from my family, I started out pursuing electronics and communications in IIT Guwahati, located near the remote and breathtakingly beautiful part of northeastern India. My desire for academic research started taking an actual shape during the years at IIT. Approximately two weeks into the first semester, I remember mentioning at an interactive student session that I intend to pursue a doctoral degree in engineering and science.

After goofing around for two years, during the sophomore summers, I worked on a small project under the guidance of Prabin K Bora. After the junior year, a three-month internship at RWTH Aachen in West Germany, under the supervision of Thomas Deserno, is the inception of a lifelong affair with signal and image processing. I learned the early lessons of academic research from Amit K Mishra and Mandar Chitre from IIT Guwahati and the National University of Singapore respectively. I am very thankful to each of

these esteemed professors for helping build a keen interest in scientific research. In Fall of 2011, I joined the University of Wisconsin and its beautiful campus situated along the banks of Lake Mendota in Madison, Wisconsin. So began a journey of doctoral studies under the supervision of Vikas Singh, and his team of collaborators from Wisconsin Alzheimer's disease research center (WADRC). I am thankful to Madhav Chitturi, who pointed out an opening for Project Assistant position at WADRC.

It will be an understatement if I say that I am indebted to Vikas for his exceedingly brilliant guidance, both as a research advisor as well as an academic mentor. There are numerous things that I learned from him — academic writing skills, the ability to choose relevant research problems, the value (and management) of collaborations, and the necessary ingredients for finishing research projects. Beyond these aspects, I will always be thankful to Vikas for being patient with me, and for the freedom he gave me regarding the types of research projects I chose. I recall many occasions when I stormed into his room, all tensed and worried, because of rejection of a manuscript for publication, or merely because an idea doesn't work. He would calm me down and sort out things very patiently. Sterling C Johnson, the associate director of WADRC, has been my single point of contact about Neuroimaging and Alzheimer's disease research. He has been an excellent mentor, and without his 'co-advising,' my work may not have taken shape. I had the opportunity to work with and learn from, Grace Wahba, Risi Kondor, Ozioma Okonkwo, Richard J Chappell, Tom Nichols, Lopamudra Mukherjee, Barbara Bendlin and Baba C Vemuri. I am grateful to my sources of financial support: NIH AG040396, NIH AG021155, NSF CAREER 1252725, NSF 1320344/1320755, UW ADRC AG033514, UW ICTR 1UL1RR025011, Wisconsin Partnership Proposal, and the many anonymous reviewers for valuable suggestions about my manuscripts.

I had terrific company, both at work and socially, during this journey of transforming from an immature, impatient, wanna-be researcher to a more sensible student. It has been a privilege sharing workspace with Chris Hinrichs, Maxwell Collins, Deepti Pachauri, Jia Xu, Wonhwa Kim, Hyunwoo Kim, Nagesh Adluru, Sathya Ravi, Seong Jae Hwang, Chris Lindner, Ronak Mehta, Felipe Gutierrez-Barragan, Yunyang Xiong, Tuan Dinh, Vishnu Lokhande, Sukanya Venkataraman and Haoliang Sung — some of whom are my co-authors. I am especially thankful to Chris Hinrichs, Deepti Pachauri and Sathya Ravi for the many fruitful discussions along the way. Outside of work, Alekhya Perugupalli, Rajesh Mukherjee, Anubhav Kushwaha, Viswa Colluru, Sriteja

Mantha, Gautam Prakriya, Sumeet Katariya, Aayushi Uberoi, Sonal Ghura and Shravan Sukumar have been a fantastic flock of drinking buddies, traveling and showtime companions, and quite simply, good friends who made my time here in Madison very memorable.

A silent traveler accompanying me on this journey has been my wife, Nitya Kalva, who I met a few years ago in Madison. I am blessed to have a fantastic partner who supports my career goals with intense zeal, shares my disappointment in failure, and keeps me grounded. Her emotional support during the latter half of my Ph.D. years is one of the very reasons I could complete this thesis. All along my journey, my parents have been very enthusiastic. A warm thanks to my mother, Kalavathi Ithapu, who believed in me achieving the goal I set forth a few orders of magnitude more than I ever did, and my father, Sarma Ithapu, who is my guiding light for getting-things-done.

Contents

List of Tables

List of Figures

Abstract

Computational methods have been rigorously studied and deployed for many advances in natural sciences, in particular, for the study of biological processes, diseases, and disorders. One such domain that benefited from computational algorithms is the study of brain diseases like Alzheimer's Disease and other forms of dementia. A natural next step for several such studies is towards developing, and testing of targeted drugs via appropriately designed clinical trials. The context of this work is the interface of well-studied classical approaches to designing efficient clinical trials, and the remarkable success of data-driven machine learning and computer vision methods in modeling the underlying disease. The questions we pose include — What goes into setting up a clinical trial? And how can machine learning methods be leveraged into improving this setup? The hypothesis for this thesis derives from the fact that *non-trivial structure* in datasets — for instance, brain scans and cognitive tests acquired for studying clinical changes in Alzheimer's disease — should provide useful information for designing clinical trials.

This thesis presents new testing, learning and inference algorithms primarily aimed at (a) designing efficient outcomes for measuring clinical trial efficacy; (b) selecting optimal study population for the clinical trial; (c) harmonizing computational outcomes and enrichers across multiple trial sites; and (d) exploring hierarchical interactions of disease markers to improve trial design. Beyond clinical trials, the proposed algorithms are applicable for fast hypothesis testing, learning neural networks from small-sized datasets, selecting/ranking hierarchically interacting entities, interpreting learned representations and other problems in machine learning and computer vision. Open-source implementations accompany each algorithm.

Nomenclature

AD	Alzheimer's Disease
MCI	Mild Cognitive Impairment
PET	Positron Emission Tomography
MRI	Magnetic Resonance Imaging
ADNI	Alzheimer's Disease Neuroimaging Initiative
WRAP	Wisconsin Registry for Alzheimer's Prevention
RapidPT	Rapid Permutation Testing
NaivePT	Naive Permutation Testing
SnPM	Statistical Non-parametric Mapping
RDN	Randomized Deep Network
rDAm	Randomized Denoising Autoencoder Marker
rDrm	Randomized Dropout Network Marker
SGD	Stochastic Gradient Descent
MMF	Multiresolution Matrix Factorization

Notation

$u, v, w, \ldots \alpha, \beta, \gamma, \ldots$	Scalars
$\mathbf{x}, \mathbf{y}, \mathbf{z}, \ldots$	Vectors
$\mathbf{U}, \mathbf{R}, \mathbf{C}, \ldots$	Matrices
U_{ij}	i^{th} row, j^{th} column entry of $\mathbf{U}$
$U_{:,j}$, $\mathbf{U}[:, j]$	j^{th} column of $\mathbf{U}$
$[m]$	$1, 2, \ldots, m$
$\|\cdot\|$	Frobenius Norm
$\nabla_{\mathbf{W}} f(\cdot)$	Gradient of $f(\cdot)$ at $\mathbf{W}$
$SO(k)$	Symmetric Orthogonal group (Order k)

Chapter 1

Introduction

Large-scale computational models, their design and deployment, played a vital role in recent advances in a variety of natural sciences including biology, physics, medicine and health care (Bower and Bolouri, 2001; Melnik, 2015; Pang, 1999). Specifically, modern medical research increasingly relies on *accumulating information* from large patient datasets to enhance our understanding of, and our ability to treat human diseases. Computational methods are ubiquitous in a wide-range of medical fields of study including neuroscience (Carmena et al., 2003; Lemm et al., 2011), cancer research (Tan and Gilbert, 2003), psychology and behavioral health (Carmena et al., 2003), cardiovascular studies (Shipp et al., 2002), genetic disorders (Medvedev et al., 2009), environmental diseases (Lampos and Cristianini, 2010), nutrition and public health (Paul and Dredze, 2011), and many more. Such a model may entail discovering non-trivial patterns from a variety of datatypes to predict a disease, model the underlying physical process that governs some detectable symptom, or propose treatments for precision medicine. Technological advances in hardware, and the ability to acquire a variety of high-dimensional high-resolution scans of the human body (e.g., magnetic resonance images – MRI, positron emission tomographic images (PET)), or high-dimensional genetic sequences (like single nucleotide polymorphisms – SNP) have contributed to new avenues in computational drug design and discovery (Szlezák et al., 2014; Groves et al., 2016).

A computational method may be statistical or *machine learning inspired*. As argued in Breiman et al. (2001), classical statistical modeling devotes significant time and resources for validating the modeling/learning assumptions. On the other hand, the approach of many modern machine learning methods is the quite the contrary — the modeling assumptions are typically by-passed, large-

sized datasets are accumulated and the data generating distributions are learned directly from the acquired data (Dietterich, 1997). Typically the goal of such models is to 'generalize' the learned patterns on unseen (future) data . The past two decades have seen the development of scalable and efficient learning models that learn from both labeled as well as unlabeled datasets (Dietterich, 2000; Zou et al., 2006; Bengio, 2009b; Liang et al., 2014; Bottou, 2010). Some such families of *supervised learning* models include multi-kernel machines, random forests and neural networks (including their modern variant, deep networks). Examples of *unsupervised learning* models include hierarchical clustering (Chandrasekaran et al., 2005; Lee et al., 2008), independent component analysis (Hyvärinen, 2013) among others. Recent advances in computer vision (the sub-area of artificial intelligence that is concerned with learning from natural images and scenes) has also contributed to the surge of machine learning in medicine, for instance, in neuroimaging with high-dimensional brain scans where more than half-a-million features/predictors are commonplace (Lemm et al., 2011; Hinrichs et al., 2011a; Plis et al., 2014; Kohannim et al., 2010).

The benefits have not been one-way though, in the sense that, one of the widely used learning models in artificial intelligence – *neural networks* – are partly inspired from our rough understanding of neuronal interactions in the brain (Minsky and Papert, 1969). The *cross-talk* between machine learning, applied statistics and multi-modal data acquisition has, in turn, contributed in advancing several fundamental research topics in artificial intelligence, including graphical models (Friston et al., 2002; Danaher et al., 2014), sparse representational learning (Rao et al., 2013; Lee et al., 2011) and high-dimensional statistics (Thornton-Wells et al., 2004; Friston et al., 1994). Although modern machine learning and computer vision methods are being extensively utilized in medical and biological research, data-hungry learning models are yet to be fully exploited within *the study, design and analysis of clinical trials*. A clinical trial is, in some sense, a culmination of years of medical research typically towards understanding the pathology of a disease of condition, and thereby designing targeted drugs or treatments to subside or cure the underlying condition. Starting from the work of Fisher on randomization (Box, 1980), and experimental design (Hall, 2007), there is a rich body of literature from applied statistics community dealing with the setup, design and analysis of clinical trials. However, as data sources become more diverse (and noisy), as the sizes of datasets increase (and may be distribute across different sites), and as the underlying disease characteristics have complex interactions, invariably the con-

duct of trials would need to exploit efficient computational techniques – taking cues from computer vision and natural language processing technologies.

The setup of a successful clinical trial involves several different aspects, and we discuss these basics in detail in Section 2.1. Consider an example scenario where the goal is to some disease in middle-aged adults and a new drug has been discovered which claims to reduce the symptoms, and eventually cure it. The hypothesis is that when administered, this drug will reduce the symptoms to a *significantly* greater extent than using an alternative medication (or no medication at all). This entails a statistical test where we monitor two groups of individuals – one group who are treated with the drug and the other non-treated ones. The test statistic then summarizes the difference between the two groups, thereby capturing the effect of mitigating the symptoms of the disease with and without the drug. Clearly, for such a test to make sense, the subjects involved in the trial should benefit from the drug i.e., they should either already have the disease, or should at least be highly probable to get the disease in the future. A typical trial is conducted on $2-4$ years, sometimes more, and during this time, it is necessary to continuously monitor the health of the individual. Lastly, trials are generally distributed across multiple sites/centers.

Either of these modules – selecting trial outcome, selecting study population, monitoring through trial time-period, and fusing inference from multiple centers – involves accumulating data and classifying the disease, or predicting the probability of disease in future (for instance, to monitor weak disease progressors during the trial). In other words, these modules of clinical trials eventually take the form of classification, regression, prediction or hypothesis testing problems. This is a key to the development of the proposed learning and inference algorithms in this thesis – the context being the *interplay* of machine learning algorithms and clinical trials design. We motivate the necessity of learning in clinical trials from three distinct, but related, attributes of computational modeling in biology and medicine. The algorithmic contributions of this thesis will then be summarized.

Structure in the data.

Starting from images of the brain (like MRI and PET), genetic sequences (like SNPs), to the medical questionnaires including electronic health records or cognitive exams, the data sources that need to be studied in clinical trials are highly *structured* (Gerber et al., 2009; Van Essen et al., 2012; McCarty et al., 2011;

Lemm et al., 2011) – see Figure 1.1. By structure, we mean that the "sources of variation" in these datasets are much smaller in number, or restricted in some sense. For example, the presence of a tumor on brain scan is physically constrained, on the other hand a cat can be present at any location in an image taken using a camera. Beyond the fact that the data themselves lie on some unknown low-dimensional manifold in some high-dimensional ambient space, there is a necessity and benefit to imposing strong priors when modeling and learning from these datasets. While statistical model building (and assumption testing) is one approach, learning such complex structures directly from large datasets (in a data-driven fashion; with minimal assumptions on the model itself) is the alternative. The success of data-driven feature and concept learning in modern artificial intelligence presents strong evidence for this alternative machine learning inspired modeling (Bengio et al., 2013a; Kim et al., 2014; Peterson et al., 2016; Brown et al., 2008). These approaches may be supervised (with such supervision provided by ground truth) or unsupervised, generative or discriminative depending on whether the data generating distribution or the target conditional is being modeled.

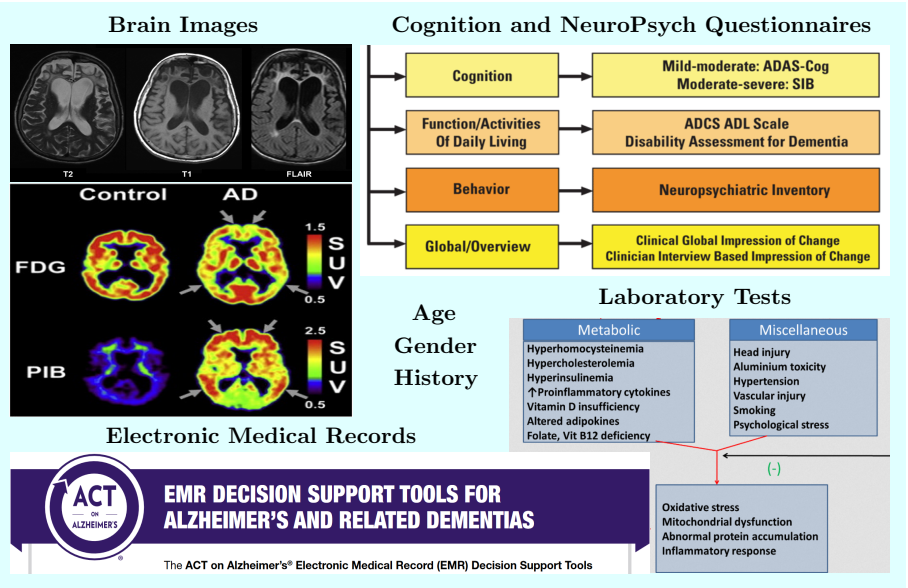


Figure 1.1: Typical medical datasets are very structured.

Complexity of concepts.

For certain types of diseases and biological processes, there are efficient and robust markers (computational or physical) – called *disease markers* – that summarize the status of the disease or the underlying pathological process. For example, high body mass index or history of stroke are strong disease markers for Type II diabetes (Klonoff et al., 2008). On the other hand, for complex disease like Alzheimer's disease – which is a form of brain dementia that causes permanent damage to cognition and memory – the scenario is not as simple. Firstly, the clinical diagnosis of the disease involves combining information a variety of patient data including the age and family history (of AD) of the individual (McKhann et al., 2011; Albert et al., 2011a; Huang et al., 2004), presence of certain types of genes (Corder et al., 1993), abnormally low scores on cognitive and effective functioning questionnaires (Albert et al., 2011b; Sperling et al., 2011; Terry et al., 1991), atypical signatures on brain scans and others. Secondly, the processes behind early disease are still being studied with many disease markers, derived from the foregoing data sources, interacting non-linearly, and inducing, what seems to be weak signals for early symptoms of the disease (Sperling et al., 2011; Dubois et al., 2016). Further, the interaction of disease markers in different stages of the disease seems to be different. This is a particularly important aspect because many trials will be targeted towards early stages of the disease – the individuals for whom neuronal death (death of neurons in the brain) is in early stages, and targeted treatments are expected to slow down the progression of the disease. Putting this all together, it is paramount to rely on computational modeling, specifically, data-driven scalable algorithms like neural networks or graphical models that learn very complex concepts. Observe that the ongoing clinical trials in medical studies like AD research need to account for the fact that the underlying disease manifestation is yet to be fully understood. Hence, understanding the disease must proceed *in tandem* with designing computational markers for disease progression – further complicating the underlying concepts that need to be modeled.

Fusing datasets.

One of the main successes of computational models in mainstream artificial intelligence, as well as in bioinformatics and medicine, has been in efficiently learning from multiple datatypes and fusing these sources towards prediction (e.g., multi-class classification (Hinrichs et al., 2011a, 2012a; Zhang et al.,

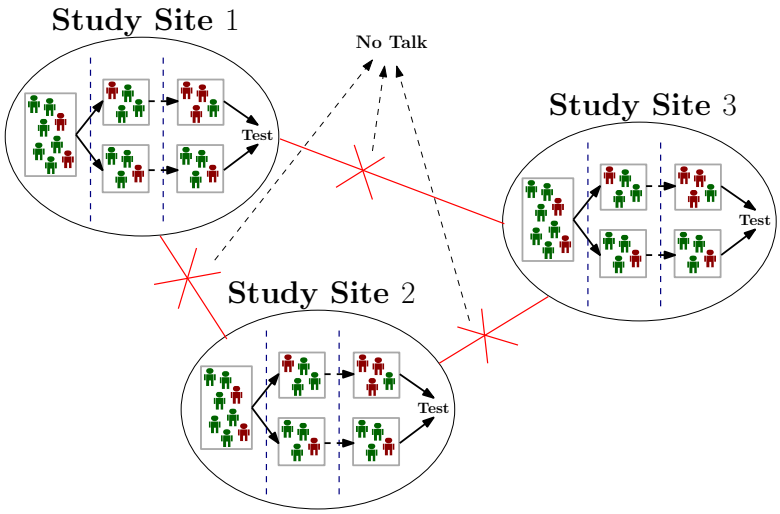


Figure 1.2: Transferable computational modeling across multiple sites.

2011a; Plis et al., 2013)). Clearly, as the structure of datasets and complexity of concepts increase, as described earlier, clinical trials conducted on diseases like AD would need to integrate a variety of data sources like images (MRI and PET), categorical scores (e.g., medical history), ordinal summaries (like cognitive exams, genetic factors) and several others. Further, the design of successful trials includes studies conducted at multiple (typically independent and international) centers/sites. These distinct sites may correspond to different data acquisition centers, different target population (e.g., Caucasians, Central Americans, Africans etc.) etc. In general, transferring data between sites is non-trivial because of the health care regulations, federal policies, logistic constraints (e.g., climate-controlled data specimens cannot be disturbed) – see Figure 1.2. Although meta analysis of multi-site trials are well studied (see Chapter 20, (Friedman et al., 2010); Mills et al. (2009), any choice of trial outcome and study population within one site should be translatable across the remaining sites. In other words, we would need computational models that are applicable for (or exploit) multi-site clinical trial design, while fusing information from multiple data sources with each site. This calls for direct integration of two modules of research studies: computational modeling aimed for understanding the disease, and computational design aimed solely at data

integration (or harmonization) (Ben-David et al., 2010; Klunk et al., 2015; Simmons et al., 2011; Jahanshad et al., 2013; McCarty et al., 2011; Van Essen et al., 2012).

1.1 Contributions

The primary concerns of this thesis work revolve around the integration of clinical trials design and machine learning algorithms. In other words, we are asking

> *What goes into setting up a good clinical trial ?* and
> *What can machine learning do to improve it ?*

To present and discuss the proposed models in detail we will restrict ourselves to clinical trials in Alzheimer's disease (AD) as the *de facto* application. And the variety of multi-modal data sources used to measure, track and predict the disease, and eventually design AD clinical trials will be the default datasets for the proposed models. Beyond clinical trials, these proposed algorithms can be generalized (and extended) to a variety of problems in modeling datasets that are highly structured and possibly data poor (a very common scenario in biology and medicine). We also note that the algorithms we discuss are generally applicable to problems in computer vision and machine learning (most of which are only loosely related to trial design). We will point out such aspects accordingly in the later chapters. In summary, all the algorithms in this thesis can be related back to the following two questions.

- How to leverage training-driven machine learning models *to design and deploy efficient multi-center clinical trials?*
- How to leverage high-dimensional and multi-modal imaging data *to set up efficient clinical trials in the early stages of AD?*

There are several different starting points for this thesis, all of which are summarized in **Chapter 2**. The first of these are the relevant background on clinical trials and AD. We begin with an extended discussion about two critical aspects in trials design that are of the most relevance to this report:

- *trial outcome:* the computational summary that measures a trial's efficacy
- *trial enrichment:* the criterion for selecting trial population

We point out an important caveat here. Beyond the choice of trial outcomes and enrichers, there are several other modules critical in the setup and design of clinical trials. These include, cross-over trials (Mills et al., 2009), multi-site

trials (Chapter 20, Friedman et al. (2010)), bandit designs (Villar et al., 2015) and others (discussed briefly in Chapter 2). As we develop the learning models, we will focus primarily on computational outcomes and enrichment criteria, and we *do not* present any alternative designs for these above well-studied constructs. The domain knowledge of the 'physics' being modeled – here this corresponds to the current (clinical and biological) understanding of all AD stages – is the other critical ingredient of this thesis. We leverage useful interpretations about, for instance, several biological and pathological disease markers. Lastly, the success of machine learning methods in *classifying* whether a given subject is healthy or diseased provides the remaining context needed for this thesis. In particular, this is the promising evidence of statistical learning methods relating early disease signatures to the changes in brain images (Zhang et al., 2011a; Hinrichs et al., 2011a). As we will discuss in much more detail in Chapter 2, the focus of clinical trials is typically in early stages of the disease (whether it is AD or not) to pick up individuals who in fact would benefit from the trial.

In Chapters 3, 4 and 5 of this thesis, learning and testing algorithms targeted toward designing efficient (single and multi-center) trial outcomes and enrichment are presented. In Chapter 6, the focus is on exploratory algorithms that mine complex structure in the data, both towards trial design and more generally for modeling inferring/interpreting non-linear interactions in the data, are discussed. Figure 1.3 visualizes all the four specific aims – with their technical and application contributions listed out, and we will walk through them one at-a-time.

The first contribution of this thesis is towards computational outcome design. In **Chapter 3**, we show that designing such an outcome, invariably, corresponds to performing a statistical hypothesis test, and with high-dimensional images or multi-dimensional covariates, the test involves multiple hypothesis correction. By observing that the classical (exact) multiple testing algorithms, like permutation testing, are very slow, we propose an algorithm called *Rapid permutation testing* (RapidPT) that achieves runtime improvements of up to $40\times$ or more at no-cost of accuracy. The key insight here is that the data on which the hypothesis test is being conducted is itself highly structured, and hence, we suggest a novel approach to *structure-driven* multiple testing algorithms – moving away from classical approaches. We also provide an analysis (theoretical and empirical) which sheds additional light on the use of matrix completion methods in this context of hypothesis tests. The proposed procedure is incorporated into the latest developer version of non-parametric statistical testing toolbox called Statis-

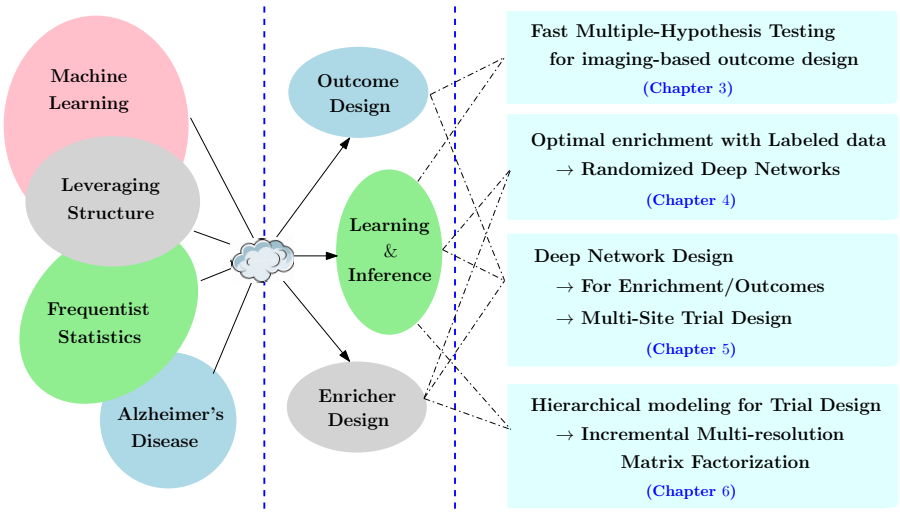


Figure 1.3: The Contributions (Specific-Aims) of this thesis.

tical Non-Parametric Mapping (SnPM, `http://warwick.ac.uk/snpm`) (SnPM, 2013). We point out that the ideas exploited in RapidPT can be applicable to most group-wise testing problems, beyond simply clinical trials outcomes design.

The proposed faster permutation testing procedure, along with existing clinical risk factors, will allow for localizing the disease signature and design good trial outcomes. In the second contribution discussed in **Chapter 4**, we start by assuming that such a trial outcome has been provided, and directly ask the question of what makes a machine learning model desirable for clinical trial enrichment, i.e., what characteristics should a learning model have, so as to use its output as a criterion for selecting the study population. We systematically tease out such desirable attributes of an *optimal* computational enricher which include

- strong discrimination power for different stages of the disease – i.e., *no approximation/modeling bias*;
- strong correlation with an existing disease outcomes or other biomarkers – i.e., strong *predictive power* of the disease; and
- small prediction variance – i.e., *small variance* on the intended outcome.

We then construct a family of learning models referred to as *Randomized Deep Networks* with precisely these characteristics. The proposed model is then

trained to output a probability score of whether a given subject should be included in the trial or not. Using a wide-variety of cognitive, executive functioning, blood-flow and imaging-summary based trial outcomes, we show that this proposed computational enricher outperforms existing state-of-the-art criteria (some of which are already being used in active/ongoing clinical trials), while producing high statistical power, with small sample size, in a simulated RCT with pre-defined effect size of the drug. We also discuss the neuroscience implications of using these models for enrichment, thereby providing a guide to the practitioner about the predictive power and *expected* sample size trends that one may achieve in observational AD trials. Beyond their use in clinical trial enrichment, the proposed models provide solutions to adapting neural network designs to small-sized datasets (typically referred to as small sample deep learning).

In **Chapter 5**, we move beyond single-site studies and ask questions about clinical trial outcome and enricher design involving multiple centers. We begin with the question of which families of network architectures are more appropriate to some given task. Here, the goal of a practitioner is to choose the network architecture most suitable for solving a given problem/application. One is then faced with a 'multitude' of architectural choices when trying to decide the specific construction that is likely to work the best. For instance, one may decide to use a network with combinations of convolutional and fully connected layers using rectified linear units where learning is performed using dropout, or prefer other variations of such a *prototypical* construction. The choice is, at least, in part, governed by the domain knowledge apart from other resource constraints including the sizes of available datasets and/or the desired convergence (or generalization) of the estimated parameters. These issues are also directly related to the design of randomized deep networks discussed in the earlier chapter. Chapter 5 is concerned with the following set of open problems, referred to as *design choice*.

- How to start from a prototypical network (like a GoogLeNet (Szegedy et al., 2014), which has combinations of convolutional and fully-connected layers) to the desired one that is 'best suited' for learning the given dataset? How does one define the best suited choice?
- How many such designs should one check for? Are there families of networks that guarantee the same level of convergence or generalization?
- How to account for resource constraints like the number of available training instances, or the maximum allowed number of parameters in the

network?

- What are the appropriate choices of non-linearities for the given task? What are the appropriate hyperparameters like dropout and denoising rates for the given dataset?

The design choice issues are more involved when models need to be constructed on datasets collected from multiple sources or acquisition sites – as in with the case of multi-site medical studies or clinical trials. This problem can often be tackled *prospectively*, e.g., within large scientific studies (Klunk et al., 2015) across multiple sites that are becoming more prevalent to better understand disease progression (Mueller et al., 2005) or evaluating the efficacy of drugs (Sperling et al., 2014). Because of privacy laws that differ between countries (e.g., Europe, United States), it may not always be possible (or may be logistically difficult) to pool/transfer data collected at different sites.

The insight here is that, if the underlying physical characteristic of the datasets are consistent, it makes sense to learn comparable deep network models across multiple (similar) datasets – without explicitly transferring the datasets. This will enable models that are *transferable* across multiple data centers ensuring that similar network models, with similar degrees of freedom, will be constructed on both data acquisition sites, respecting geographical data transfer constraints. The proposed solution strategy involves several modules: We first build an analysis framework for multi-layer deep networks, tying the architectural and learning hyper-parameters to stochastic gradient optimization. With this framework in hand, one can then compute the families of different deep networks that have the same level of gradient convergence. Posing this as a model selection problem, given a pre-specified level of convergence, we propose systematic design strategies that will *estimate* the network architecture for the given task. These strategies also rely on data statistics, and hence, the resulting best choice of networks will be the same across multiple sites (whenever the data meta-statistics are the same across sites – this is almost always true because the underlying physical characteristic is shared across centers).

In the earlier contributions, we posed the design of trial outcomes and enrichers as a selection procedure based on the output of some appropriately trained supervised machine learning model. In the final contribution from **Chapter 6**, we instead take an alternate approach of selecting the most informative *or unique* subjects and features with respect to the underlying data generating distribution. This involves estimating each subject's *leverage* when designing a trial outcome or enrichment criterion, and the key is to extract

non-trivial structures from subject-by-subject (or feature-by-feature) similarity matrices. We begin by empirically motivating that the existing approaches to model parsimonious (hierarchical) structure in symmetric matrices is flawed. The defacto constructs like sparsity, low-rank or a decaying eigen-spectrum cannot account for the "block" or cluster-like structures inherent in real world datasets, including those encountered in medical research and AD. We first propose a scalable multi-scale factorization on (large and dense) symmetric matrices by putting together two fundamental data decomposition approaches – wavelets and matrix factorization. The proposed algorithm called *incremental multiresolution matrix factorization* estimates a set of sparse rotations from the underlying matrix, eventually compressing it to a block-diagonal form. The proposed algorithm has several applications, both specific to clinical trials design and beyond:

- Generalizing statistical leverage scores i.e., generalizing unsupervised importance sampling (where the goal is to select an instance or feature that is most informative in some sense);
- Designing hypotheses based on lower and higher order interactions within the dataset – which is also related to building null hypotheses of statistical tests;
- Visualizing hierarchical structure within the data beyond tree-based (agglomerative) clustering or grouping; and
- Interpreting non-linear learning models using covariance (or other similarity) of learned representations.

Chapter 2

Background

We begin with some background on clinical trials and Alzheimer's disease.

2.1 Clinical Trials

The authors of (Friedman et al., 2010) define a clinical trial as a prospective study comparing the *effect and value of an intervention or a treatment against a control or placebo* in human beings. A group of individuals are given an intervention or a treatment, who are then compared to the controls (Cook and DeMets, 2007). For example, the study's goal may be to figure out if a newly discovered drug will cure cancer. Here, the intervention or treatment will correspond to administering the drug to an individual, and a control would be a person who will *not* be treated. An intervention could entail administering a drug or vaccine, it may impose changes in daily-life (e.g., dietary choices, supplements, scheduled exercises), or more generally it may correspond to implanting medical devices and monitoring the individual's health. A clinical trial can rely upon multiple combinations of interventions if needed, depending on the goal.

The evolution of modern trials dates back to the eighteenth century (Bull, 1959), and in general a clinical trial has 4 phases. Phase I assesses the safety of the intervention. Phase II tests the efficacy of the treatment on a small scale involving several hundred individuals, and Phase III builds upon this by testing the treatment's goodness in a large population. Both these phases typically involves multi-site trial designs. Phase IV involves post marketing studies after a drug or device has been approved for consumer sale. A properly planned and executed clinical trial is a powerful experimental technique for assessing the effectiveness of an intervention. Apart from the standard books on statistical methods in clinical trials, (Friedman et al., 2010) and (Cook and DeMets, 2007),

the authors of (Donner, 1984; DerSimonian and Laird, 1986) provide detailed analysis of the setup and procedures involved in conducting clinical trials in practice.

Randomized Controlled Trial (RCT)

Randomized controlled trials are comparative studies where the assignment of the subject to a group – intervention or control – is determined by the formal procedure of randomization (Friedman et al., 2010). Randomization, in the simplest case, is a process by which all participants are equally likely to be assigned to either of the groups. Such randomization removes the potential of selection bias in the allocation of participants, thereby producing comparable (and reproducible) results across multiple independent repetitions of the trial (with same drug but different population). In other words, random assignment ensures that the controls are not biased representers of intervened (Chapter 5, (Friedman et al., 2010); Chapter 5, (Cook and DeMets, 2007)). Further, it guarantees the validity of statistical tests of significance that measure how different the intervention subjects are compared to controls (see (Donner, 1984) and others for more details). RCT is often considered as the gold standard in trial design, and Figure 2.1 shows the work-flow of RCT. If at the end of the trial, the controls and intervention groups differ *significantly*, then the intervention or treatment had some non-trivial affect on the population of interest, which calls for more analysis and interpretation of the results.

Trial Outcome: Each RCT must have a primary question, and invariably, it is in the form of *testing a hypothesis*. This is because most of the time the intervention is postulated to have a particular *outcome* which, on the average, is different from the outcome on the control group (Chapter 3, (Friedman et al., 2010); Chapter $2-3$, (Cook and DeMets, 2007)). Such a trial outcome may correspond to a clinical event, or change in disease intensity towards improving health, or reducing symptoms prescribed a priori to the trial etc – essentially a *reliable* disease marker.

Trial Enrichment: The study population is the subset of individuals with the condition or characteristic of interest (Chapter 10, (Friedman et al., 2010); Chapter 4, (Cook and DeMets, 2007)). For instance, a trial that tests the efficacy of some intervention for smoking cessation should be conducted on those

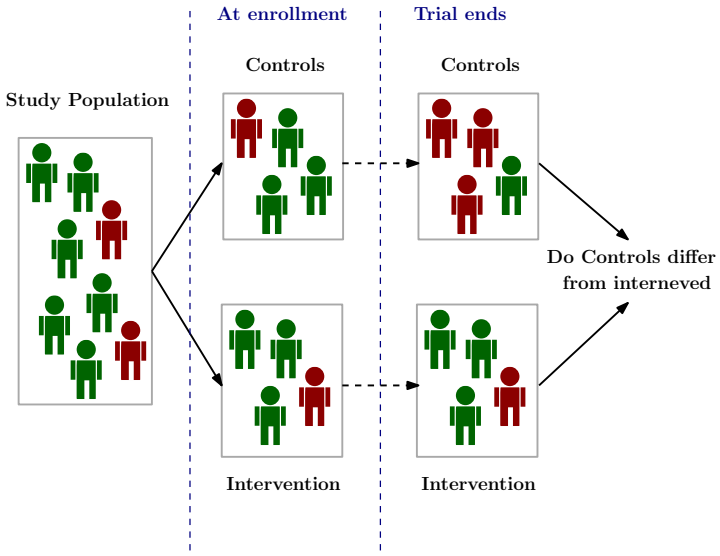


Figure 2.1: Randomized Controlled Trial Work-flow

people who, in fact, do smoke. This trial eligibility criterion is referred to as an *enrichment criterion* i.e., the trial is enriched to include those subjects who in turn would be beneficial from it. Once this study population is finalized, the RCT then *randomly* assigns each subject to either of the arms.

The Landscape of Clinical Trials: Setting up a successful clinical trial involves a variety of other aspects beyond the choices of the trial outcomes and the study population – although, to begin with, randomization of intervention and treatment groups, and an enrichment criteria (for study population) are sufficient to *start* the trial. The design and analysis of clinical trials is backed up by more than half-a-century of rich literature in statistics and epidemiology communities, starting from the introduction of randomization by R A Fisher in the monograph titled Statistical Methods for Research Workers (Fisher, 1934). A few of the other statistical constructs – beyond outcome design and study population – vital for setting up a trial include blindness (Chapter 7, (Friedman et al., 2010)), survival analysis (Chapter 15, (Friedman et al., 2010); Chapter 7, (Cook and DeMets, 2007)), interim analysis (Chapter 17, (Friedman et al., 2010); Chapter 10, (Cook and DeMets, 2007)). Qualitative aspects like baseline assessment (Chapter 9, (Friedman et al., 2010);), quality-of-life analysis

(Chapter 11, (Friedman et al., 2010); Chapter 9, (Cook and DeMets, 2007)) have received tremendous attention. Crossover and multi-center trials that boost statistical power by distributing the single trial across multiple sites are well studied (Chapter 20, (Friedman et al., 2010); (Mills et al., 2009)). More subtle, but practical, issues including missing data (Chapter 11, (Cook and DeMets, 2007)), longitudinal analysis (Chapter 8, (Cook and DeMets, 2007)) and eventually interpreting the results (Chapter 19, (Friedman et al., 2010)) have received attention. More recently, there have been advances in Multi-armed bandits (Villar et al., 2015) and adaptive removal/selection (Simon and Simon, 2013) for monitoring the conduct of an ongoing trial. In general, these statistical constructs study and analyze the trial specifics *after* the trial design has been specified i.e., after the clinicians choose the desired outcome and study population. This caveat is important as we present the contributions of this thesis in Section 1.1.

2.2 Alzheimer's Disease

Dementia is a general term for a decline in mental ability severe enough to interfere with daily life[1]. Alzheimer's disease (AD) is a type of dementia that causes problems with memory, thinking and behavior (Association, 2015; Ferri et al., 2006). It is a progressive brain disorder that damages and eventually destroys brain cells, leading to memory loss and changes in thinking and other brain functions [2][3]. The disease usually develops slowly during the late middle age, and gradually gets worse as brain function declines and brain cells eventually wither and die. Ultimately, this neural death is irreversible and AD is fatal. Currently, there are no known cures for the disease, but several treatments are under study which may slowdown the disease progression and delay the disease onset. In the United States, 1 in 9 adults aged 65 and older have the disease, and by 2050 this number is expected to triple with an approximate 15 million Americans having some form of AD (Association, 2015; Ferri et al., 2006). AD is the most common form of dementia among people over the age of 65, and as of 2017 more than 5.3 million Americans are affected by it.

Although decay of cognition is the primary signature of the disease, the greatest known risk factor for AD is *advancing age* (Bateman et al., 2012). Family

[1] http://www.alz.org/what-is-dementia.asp
[2] http://www.alz.org/Alzheimer's_disease_what_is_alzheimers.asp
[3] http://www.alz.org/research/science/alzheimers_research.asp

history (a parent, child, brother or sister having AD) and Apolipoprotein-E4 (APOE4) genetic variation are the other strong risk factors (Huang et al., 2004; Dubois et al., 2007). There is no single test to diagnose AD. A careful evaluation of medical history, mental status tests, physical and neurological exam, and a combination of blood tests and brain imaging (to rule out other forms of dementia) is needed to diagnose an older adult with AD [4] (Bateman et al., 2012; McKhann et al., 2011). AD participants are typically evaluated via NINCDS/ADRDA criterion for probably AD [5]. We briefly discuss the stages involved in a individual's health while progressing to dementia, followed by the variety of data sources that are acquired to capture this progression of the disease.

Early signatures of AD

Mild Cognitive Impairment: A milder form of dementia that precedes full-blown AD is referred to as *mild cognitive impairment* (MCI) (Albert et al., 2011b; Petersen, 2007). An MCI subject should have reported a subjective memory complaint, with no serious cognitive dysfunction, either autonomously or via a clinician or informant. Levels of MCI – *early or late* are determined based on Wechsler Memory Scale Logical Memory II. Annually about $13 - -15\%$ of MCI sufferers convert to AD on average, however some remain as stable MCI without ever converting. Thus MCI is usually, but not always, a precursor to AD. Hence, there is interest in discriminating which MCI subjects will progress, and which subjects will not, since any discriminative model (computational or otherwise) which can do so can potentially highlight factors which differentiate AD from normal aging. Recently, evidence from longitudinal studies conducted across several international AD cohorts suggests that accumulation of plaques and tangles in the brain – that typically precedes even the subtle memory complaints – might be responsible for eventual brain tissue shrinkage and loss of connections among brain cells (Association, 2015; McKhann et al., 2011). During the MCI, and even in the early MCI, stages this resulting brain damage can be captured primarily via structural magnetic resonance images (MRI) and functional positron emission tomographic (PET) images. The seminal papers by the authors in (Bateman et al., 2012; Pike et al., 2007) have shown a complex interaction between such imaging signatures and the several AD risk factors

[4] `http://www.alz.org/alzheimers_disease_diagnosis.asp`

[5] National Institute of Neurological and Communicative Disorders and Stroke (NINCDS) and the Alzheimer's Disease and Related Disorders Association (ADRDA)

implying that beyond cognitive tests, direct evidence (strong signal) from brain images can be suggestive of mild to moderate MCI.

Preclinical AD: An important observation, reported in multiple scientific publications, is that the weak disease signatures on MRI and PET may still be observed even when the cognitive and effective functioning tests are reasonably mild – or even normal or *asymptomatic* (Mueller et al., 2005; Albert et al., 2011a). In general, such signatures correspond to mild atrophy (loss of tissue in grey matter i.e., lack of signal) and mild Amyloid burden (accumulation of plaques i.e., increase of signal). Preclinical AD refers to the full spectrum from completely asymptomatic (i.e., completely normal/healthy – denoted by CN) individuals with some biological evidence of Alzheimer's to individuals manifesting subtle cognitive decline but who do not yet meet accepted clinical criteria for MCI. The biological evidence, referred to as a *biomarker* corresponds to presence of plaques, tangles and Amyloid burden in the brain, all of which can be captured via a variety of PET images or cerebrospinal fluid (CSF) measurements (see the discussion in Section 2.3). In this preclinical stage, the atrophy in grey matter is extremely mild. Several retrospective analyses have recently suggested that the signatures, albeit mild, first manifest on brain imaging data (see discussion from Section 2.3), prior to the subjects showing any form of cognitive decline what-so-ever (Pike et al., 2007; Mueller et al., 2005; Sperling et al., 2011; Jack et al., 2013). Since AD leads to permanent brain damage, experts have argued that detecting and tracking the disease during the early stages of MCI, and ideally in the preclinical stage itself, is vital for designing targeted drugs and interventions for delaying the cognitive decay (Sperling et al., 2011; Dubois et al., 2016).

2.3 Data Sources in AD studies

We now briefly summarize the list (and landscape) of data types/sources that are relevant for AD research and analysis. Apart from high-dimensional brain images – which are the primary data types analyzed and exploited extensively for the proposed models – several risk factors, physical exams about cognition and memory, and biomarker summaries will be the *default covariates* for the proposed algorithms. Please see (Albert et al., 2011b; McKhann et al., 2011; Sperling et al., 2011; Weiner et al., 2015; Petersen, 2007; Weiner et al., 2013)

and other appropriate references[6] [7] [8] there in for complete details about these disease markers, data types and risk factors.

High-dimensional Brain Imaging. Brain images have been one of the most important data sources for studying AD, including other forms of brain disorders. Almost all of the advances in computational biomarkers during these early stages rely on the imaging data (Lemm et al., 2011; Hinrichs et al., 2011a; Plis et al., 2014; Kohannim et al., 2010). T1-weighted MRI, T2-weighted fluid-attenuated inversion recovery (FLAIR) MRI, and diffusion tensor imaging (DTI, also known as diffusion MRI) are the widely studied brain scanning mechanisms for capturing structural changes in grey matter tissue and white matter tracts (Frisoni et al., 2010; Jack et al., 2008; Corouge et al., 2004). Functional MRI is also recently being studied, however to a smaller extent (He et al., 2007). PET scans summarize the functional state of the brain by capturing amyloid plaque accumulation. Fludeoxyglucose 18 (FDG) PET played an important role during the early 2000s when [F-18]Florbetapir amyloid $\beta 42$ PET was still being developed (Rowe et al., 2008). Both capture signs of decay in neuronal activity in brain regions associated to memory and cognition (Chetelat et al., 2003). Recently, there is evidence suggesting that Pittsburgh compound B (PIB) and Tau PET scans capture the very early signatures of neurofibrillary tangles in the brain (Jack et al., 2009; Klunk et al., 2004). Lastly, these imaging sources are represented as 3D volumes – each of which comprises of few thousand signals corresponding to the locations inside the brain. One can think of these as stacks of 2D matrices arranged as one 3D volume, and the locations correspond to *voxels* (i.e., pixels in 3D space). Figure 2.2 example 2D slices of MRI, FDG PET and PIB PET, corresponding to healthy, MCI and completely AD subjects.

Risk Factors. The single greatest risk factor for developing sporadic AD is advancing age. Most people with the disease are 65 and older. One in nine people in this age group and nearly one-third of people age 85 and older have the disease. The second strongest risk factor is Family history. Those who have a parent, brother or sister with Alzheimer's are more likely to develop AD. The risk increases if more than one family member has the illness. Several known

[6] http://adni.loni.usc.edu/methods/documents/

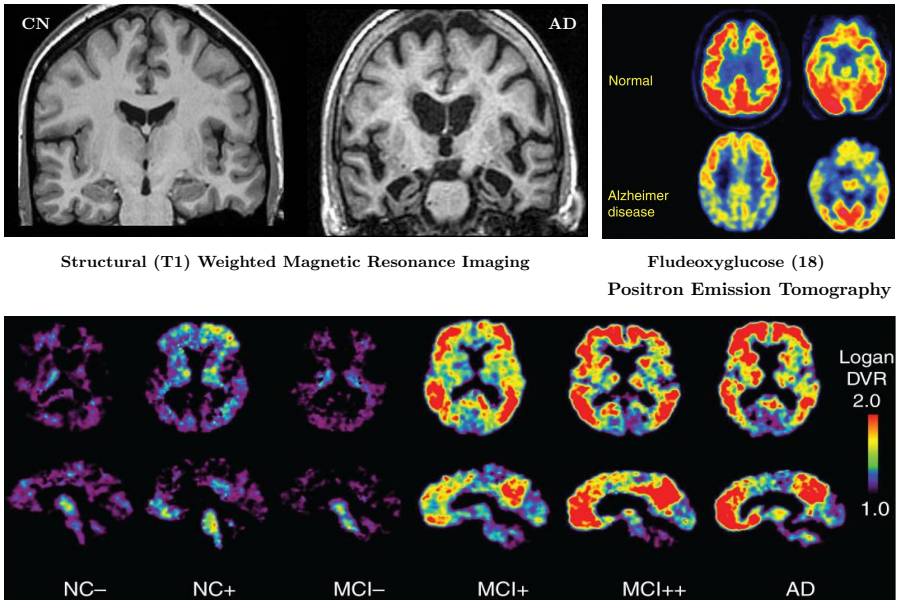
[7] http://www.alz.org/research/science/earlier_alzheimers_diagnosis.asp

[8] https://adni.loni.usc.edu/wp-content/uploads/2008/07/adni2-procedures-manual.pdf

genes have also been linked to AD. Apolipoprotein E-e4 (APOE4) is the first such gene variation found to increase risk of Alzheimer's and remains the risk gene with the greatest known impact (Corder et al., 1993). Apart from APOE4, Amyloid precursor protein (APP), Presenilin-1 (PS-1) and Presenilin-2 (PS-2) are also linked to AD [9].

Cognitive and Neuropsychiatric Tests. Since the primary consequence of AD is problems with memory and cognition, several classical, and celebrated cognitive testing mechanisms from psychometric analysis literature have been used to both define and diagnose the disease stage as well as track changes in subject's progression. Neuropsychological assessment also aids in the diagnosis of the disease by objectively establishing cognitive impairment from such standardized tests (Albert et al., 2011b; McKhann et al., 2011; Sperling et al., 2011; Terry et al., 1991). A few of the widely used cognitive and neuropsychiatric tests

[9] https://www.alz.org/alzheimers_disease_causes_risk_factors.asp



Structural (T1) Weighted Magnetic Resonance Imaging

Fludeoxyglucose (18) Positron Emission Tomography

Pittsburgh compound B Positron Emirrion Tomography

Figure 2.2: Slices/snapshots of MRI, FDG and PIB PET brain scans.

include: [10] [11] Mini-mental state exam (MMSE), Alzheimer's Disease Assessment Scale (ADAS), Montreal Cognitive Assessment (MOCA), Rey Auditory Verbal Learning Test (RAVLT), neuropsychological summary score for Memory (PsyMEM), neuropsychological summary score for Executive Function (PsyEF), everyday Cognition (eCOG), Geriatric Depression Scale (GDS), Functional Assessment Questionnaire (FAQ) and Neuropsychiatric Inventory Questionnaire (NPI). Each of these tests/questionnaires/scales comprise of a list of questions targeted towards memory and cognition of the participant. The responses to each of the question is scored (on some reasonably predefined scale, e.g., $1-5$ or Yes/No), and the overall score is a combination individual question-wise scores (simply a sum, or more elaborate). For instance, MMSE lies between 0 and 30, where 30 is CN and smaller scores imply memory impairment. The cognitive and functional performance is typically summarized using Clinical Dementia Scale sum-of-boxes (CDR-SB) going from 0: healthy to $\geqslant 3$: severe dementia.

Biomarkers. A biomarker (i.e., a biological marker) is objectively measured and evaluated as an indicator of normal biological processes, pathogenic processes or pharmacological responses to a therapeutic intervention (Mayeux, 2004). Three biomarkers have been well-established and validated internationally to diagnose AD in cerebrospinal fluid (CSF): β-amyloid$(1--42)$ [A$\beta(1--42)$], total τ and phospho-τ-181 (Humpel, 2011; Blennow et al., 2010). There is now a consensus that only the combination of these three CSF biomarkers significantly increases the diagnostic validity for sporadic AD, which yields a combined sensitivity of $> 95\%$ and a specificity of $> 85\%$ (Humpel, 2011). The primary signature of early dementia is in fact the deposition of β-Amyloid as captured by $A\beta$ PET (Pike et al., 2007; Rowe et al., 2008) and Pittsburgh B (PiB) PET images (Klunk et al., 2004), thereby using high-dimensional brain images as a biomarker source as well. Apart from these data types, a variety of vascular summaries like systolic and diastolic cholesterol levels, body mass index, eating and sleeping habits etc. are also used as covariates in several studies. In Figure 2.3 we show the summary of the disease stages as captured by the above discussed imaging, cognition and biomarker summaries.

[10] `https://adni.loni.usc.edu/wp-content/uploads/2008/07/adni2-procedures-manual.pdf`

[11] `http://adni.loni.usc.edu/methods/documents/`

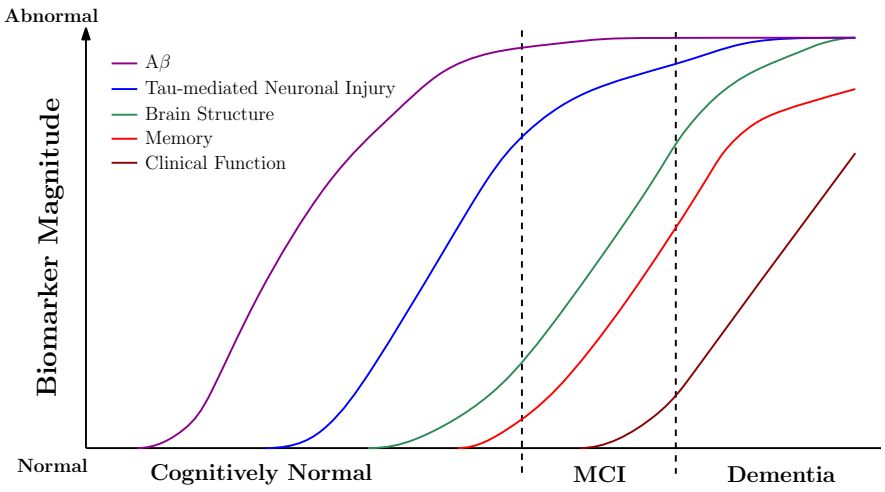


Figure 2.3: Change in biomarkers as Alzheimer's progresses.

Alzheimer's Disease Neuroimaging Initiative (ADNI)

ADNI is a multisite study that aims to improve clinical trials for the prevention and treatment of AD – `http://adni.loni.usc.edu/`. The primary goal is to track the progression of the disease using biomarkers so as to assess the brain's structure and function over the course of the different disease states. There were multiple different phases for the ADNI study – ADNI 1, ADNI GO, ADNI 2 and ADNI 3 (`http://adni.loni.usc.edu/study-design/`) – the first phase ADNI 1 began in 2004 under the leadership of Dr. Michael W. Weiner. It was launched by the National Institute on Aging (NIA), the National Institute of Biomedical Imaging and Bioengineering (NIBIB), the Food and Drug Administration (FDA), private pharmaceutical companies and non-profit organizations, as a 60 million, 5-year public-private partnership. It enrolled 200 elderly CNs, 400 MCI and 200 AD subjects. For each such enrolled participant, a variety of the data sources from Section 2.3 are acquired. The cohort sizes of ADNI GO, 2 and 3 were much bigger with some participants from earlier phases carried forward for continued monitoring along with new participants being enrolled.

ADNI 2 comprises of > 1300 participants, and the enrollment includes people between the ages of 55 and 90, recruited at 57 sites in the United States and Canada. All the acquired data is shared across research centers. Many partici-

pants are being monitored over almost a decade (with interval of six months or more) provided highly sensitive (and extensive) medical history – a vital tool for understanding AD progression. The imaging protocols are standardized. MRI images are MP-RAGE/IR-SPGR from a 3T scanner. PET images are 3D scans consisting of four 5-minute frames[12] [13] from 50 to 70 minutes post-injection for AV45 PET, and six 5-minute frames from 30 to 60 minutes post injection for FDG PET. Cognitive tests typically involve a moderator conducting the examination on some subject in an isolated environment (a typical interview room). For further details about the distinct goals of each phase, the complete list of data sources acquired and ongoing investigations (and participant research groups), please refer to `http://adni.loni.usc.edu/`.

Wisconsin Registry for Alzheimer's Prevention (WRAP)

WRAP is a longitudinal observational cohort study enriched with persons with a parental history of probable AD (Johnson et al., 2017). Established in 2001, it is the world's largest such study on adults with family history of AD. The study is part of Wisconsin Alzheimer's Institute (WAI) and Wisconsin Alzheimer's Disease Research Center (WADRC). 1550 individuals have been followed for the past 16 years on their biological, health and lifestyles factors that affect the disease. The study goals are to identify early cognitive decline, and to characterize midlife factors associated with such decline and the contributing underlying biomarkers of AD and related pathology. The data's richness includes lifestyle, physical activity, genetics, and metabolomics, apart from several of the high-dimensional imaging sources as well as a battery of cognitive scores and biomarkers. For further details about the goals and the complete list of data sources and recent insights, please refer to `http://www.wai.wisc.edu/research/whatiswrap.html`. For the most part we will use cognitive tests acquired within the WRAP cohort in our evaluations. Specifically the list of tests including their abbreviations are shown in Table 2.1. All these scores test a combination of executive function, memory and other aspects of cognition relevant to dementia.

[12] `http://adni.loni.usc.edu/methods/documents/mri-protocols/`
[13] `http://adni.loni.usc.edu/methods/pet-analysis/pet-acquisition/`

Abbreviation	Score Name
iqdspf	Digit span forward
iqdspb	Digit span backward
iqlnsr	Letter-Number Sequencing
cfl	Letter Fluency
bnt	Boston Naming Test
avlt-tot	AVLT total trials $1-5$
AVLT-del	AVLT delayed recall
stroopcw	Stroop color-word interference
trla	Trails A time
trlb	Trails B time
BVMT-tot	BVMT total trials 1-3
BVMT-del	BVMT delayed recall

Table 2.1: Cognitive scores acquired in WRAP study.

Data Preprocessing.

The cognition scores and several of the risk factors and including biomarker summaries are ordinal, and in certain cases nominal, and these summaries do not require any significant pre-processing before using them in a learning algorithm. The imaging data on the other hand needs to go though a long pipeline of pre-processing involving several *benchmarked* steps using widely used softwares in neuroimaging research studies – Statistical Nonparametric Mapping (SPM; `http://www.fil.ion.ucl.ac.uk/spm/`) (Ashburner et al.). The first step is to register all the brain images to a common coordinate space. This is because human subjects have widely-varying head sizes, and before passing the images to a learning algorithm meaningful set of features needs to be generated by establishing point-wise anatomical correspondences. This is done by one of the two non-linear registration pipelines: voxel-based morphometry (VBM) (Ashburner and Friston, 2000, 2001) or tensor-based morphometry (TBM) (Hua et al., 2008). The two approaches have different objective in terms of computing the non-linear warp.

VBM requires only global alignment of brain structures with limited degrees of freedom in transformation, whereas TBM performs best when the registration is highly deformable and can achieve higher registration accuracy. Both VBM and TBM can be followed by a Gaussian smoothing step if needed. A notable alternative is Region of Interest (ROI) analysis, in which a specific brain region is segmented, either manually or automatically, from which summary statistics are extracted. The ROIs themselves come from a "predefined" dictionary of

tessellation of the brain space (Tzourio-Mazoyer et al., 2002). Each ROI typically corresponds to anatomically located voxels, and the accumulated signal may be the mean or mode of voxel-wise values within this region. ROI analysis has the advantage of boosting signal-to-noise ratio, but it may lose information through aggregation into summary statistics, and by ignoring whole-brain features. For each brain image, the output of the processing pipeline are few hundreds of thousands of voxel-wise scalars, or much smaller number of ROI summaries (both of which are positive signals).

2.4 Machine Learning for AD Modeling

In the past decade, several authors have proposed machine learning methods for classifying the different stages of AD, specifically for discriminating MCI subjects from completely demented ones, achieving $\approx 90\%$ accuracy of deciding whether a given subject is MCI or AD (Hinrichs et al., 2011a; Suk and Shen, 2013; Davatzikos et al., 2011; Zhang et al., 2011b). Even classifying late MCI from early MCI and CN subjects has been well studied achieving $> 85\%$ accuracy. Although there is no single model-of-choice, several of these studies rely on random forests (Lebedev et al., 2014), singe or multi kernel machines (Klöppel et al., 2008; Hinrichs et al., 2011b), generalized linear models (Davatzikos et al., 2011) and more recently deep networks (Suk and Shen, 2013). Alternatively, several regression methods have also been proposed in predicting the cognitive decline in MCI (Teipel et al., 2007; Tatsuoka et al., 2013). Typically, the inputs for these methods include some combination of imaging modalities (MRI and PET), risk factors (age, history, APOE et.) and possibly other bio-markers. The output is, in general, the disease diagnosis measured on the scale of CDR-SB. On the other hand, variable selection methods have also been proposed for localizing the voxels (3D pixels) on the brain scans that correlate to the presence of AD and late MCI (Kim et al., 2014; Hinrichs et al., 2011a).

The more difficult question of predicting early decline, and the landscape of early MCI and preclinical AD (see Section 2.2 for description about these regimes) has received less attention. This is because of weak signal-to-noise ratio compared to post MCI stage, as well as the complex interactions of the risk factors like age, family history, APOE4; with demographic (gender, education etc.), vascular (body-mass-index, cholesterol, other incipient vascular diseases etc.) and other biomarkers including plaques, tangles, cumulative Amyloid burden makes early detection very difficult. A bulk of recent AD research is

focused on understanding the interactions and their affect on early decline (Dubois et al., 2016; Bateman et al., 2012; Huang et al., 2004; Vemuri et al., 2009; Pike et al., 2007; Lopez et al., 2006), and possibly trace out the weak signatures of the disease for subjects in their early 60s using all the possible data sources (see Section 2.3) (Mosconi et al., 2010). The interaction of such multi-modal information sources invariably calls for robust statistical and machine learning algorithms to efficiently *fuse* the data, and the first such studies exhaustively studying early MCI and preclinical AD include (Hinrichs et al., 2011b; Zhang et al., 2011a).

Unlike in the later stages of AD where voxel-wise learning algorithms i.e., machine learning models that directly operate on the 3D voxel space, the models proposed for preclinical AD typically work with ROIs (Edison et al., 2007) – primarily to boost the signal. These computational methods have not only resulted in better understanding of the manifestation of AD, but also have provided new insights into the disease signature which may have been difficult to capture otherwise. Several representation learning algorithms are also being studied in the context of preclinical AD (Fennema-Notestine et al., 2009; Teipel et al., 2013), and in general, designing learning algorithms for preclinical AD studies involves high degree of feature selection and pre-processing on the datasets because of weak signal-to-noise ratio. Nevertheless, there is no consensus if the complex interactions of all bio-markers is the same across all the stages, implying that the computational methods designed for late-MCI and AD stages may not directly be applicable to early-MCI and preclinical stages. This is still an open question, and so is *tracing back* early cognitive decline from MCI to healthy middle age – a vital component if one intends to design interventions *before* any form of mild dementia manifests in the subject. This broad role and success of machine learning in AD provides promising evidence, as well as the necessary context, for the impact of the contributions from this work.

Chapter 3

Fast Hypothesis Testing for Outcome Design

3.1 Introduction

Suppose we have completed a RCT of a promising new drug targeted towards AD's neurodegenerative symptoms, on older middle-aged adults. Among the many biomarkers, cognitive and effective functioning summaries are the typical choices for AD trial outcomes (see the definition of trial outcome from Section 2.1). In addition to assessing improvements in standard cognitive outcomes (see the discussion from Section 2.2 and 2.3), the purported treatment effects will also be assessed using brain imaging data. This is particularly the case with trials conducted in early stages of AD, because even if the drug does induce variations in the cognitive symptoms, the brain changes are observable *much earlier* in the imaging data (as discussed in Section 2.2). Hence we may be interested in cognitive as well as brain imaging based trial outcomes. Evaluating such an outcome amounts to performing a statistical test between the two arms of the trial, and assessing group differences.Computationally, this is simply a statistical test on the desired trial outcomes – see the discussion from Section 2.1. And clearly, the efficacy hinges on whether the outcome we chose is *sensitive* enough to pick small signal differences between the two arms. In designing machine learning algorithms for clinical trials, this is the first stop – *efficiently evaluating trial outcomes.*

Measuring group differences is not specific to trial outcomes, for instance, consider a second scenario where we have completed a neuroimaging research study of a particular controlled factor, such as genotype, and the interest is to evaluate *group-wise* differences in the brain images: to identify which regions are affected as a function of class membership. In order to localize the effect under investigation (i.e., treatment or genotype; or diseased or healthy), we

then have to calculate a very large number of univariate test statistics – typically of several million *voxel-wise* statistics if high-dimensional brain imaging data is being studied, much less if we are interested in other biomarkers. The choice of statistic depends on the hypothesis being tested. For example, these include group-contrast t-statistics for testing two groups, the F statistic primarily used in analysis of variance (ANOVA) testing, Pearson's correlation statistic used in association studies, or the χ^2 test of dependence between variables. We will restrict ourselves to the two group setting with t-statistics and the high-dimensional imaging data for the remainder of this chapter.

In some voxels, it may turn out that a group-level effect has been indicated, but it is not clear right away what its true significance level should be, if any. As one might expect, given the number of hypotheses (hundreds of thousands to millions), multiple testing issues in this setting are quite severe, making it difficult to assess the true Family-Wise Type I Error Rate (FWER) (Westfall and Young, 1993). If we were to address this issue via Bonferroni correction (Bland and Altman, 1995), the large number of separate tests implies that certain weaker signals will almost certainly never be detected, even if they are real. This directly affects studies where detecting weak signals is critical – like neurodegenerative disorders in which atrophy proceeds at a very slow rate and the therapeutic effects of a drug is likely to be mild to moderate anyway. This is a critical bottleneck which makes localizing real, albeit slight, short-term treatment effects problematic. Already, this restriction will prevent us from using a smaller sized study (fewer subjects), increasing the cost of pharmaceutical research. In the worst case, an otherwise real treatment effect of a drug may not survive correction, and the trial may be deemed a failure.

Bonferroni versus true FWER threshold. Observe that theoretically, there *is* a case in which the Bonferroni corrected threshold is close to the true FWER threshold: when point-wise statistics are i.i.d.. If so, then the extremely low Bonferroni corrected α-threshold crossings effectively become mutually exclusive, which makes the Union Bound (on which Bonferroni correction is based) nearly tight. However, when variables are highly *dependent* – and indeed even without smoothing there are many sources of strong non-Gaussian dependencies between voxels, the true FWER threshold can be much more relaxed, and it is precisely this phenomenon which drives the search for alternatives to Bonferroni correction. Thus, many methods have been developed to more accurately and efficiently estimate or approximate the FWER (Li and Ji, 2005a; Storey and Tibshirani, 2003; Finner and Gontscharuk, 2009; Leek and Storey, 2008),

which is a subject of much interest in statistics (Clarke and Hall, 2009), machine learning (García et al., 2010), bioinformatics (Ge et al., 2003), and neuroimaging (Nichols and Hayasaka, 2003a). Observe that, the many univariate statistics may be indistinguishable from a Gaussian, but it is their *interdependencies* that are strong, and in general non-Gaussian. Alternatively, if these interdependencies are *fully* captured by the covariance matrix, then the strategy would be to change the eigenbasis followed by univariate non-parametric statistical testing with Bonferroni correction.

Permutation Testing

A commonly used method of directly and non-parametrically estimating the FWER is Permutation testing (Nichols and Hayasaka, 2003a; Singh et al., 2003), which is a method of sampling from the Global (i.e., Family-Wise) Null distribution. Permutation testing ensures that any relevant dependencies present in the data carry through to the test statistics, giving an unbiased estimator of the FWER. If we want to choose a threshold sufficient to exclude *all* spurious results with probability $1 - \alpha$, we can construct a histogram of sample maxima taken from permutation samples, and choose a threshold giving the $1 - \alpha/2$ quantile. As opposed to parametric hypothesis testing schemes (K.J. Friston, 1995; Worsley et al., 1992, 1996), nonparametric permutation tests (Holmes et al., 1996; Nichols and Holmes, 2002) can provide exact control of false positives while making minimal assumptions on the data. There is consensus that it is perhaps one of the most commonly used non-parametric statistical techniques in medical imaging studies (Arndt et al., 1996; M. Halber and Heiss, 1997; Holmes et al., 1996; Nichols and Holmes, 2002; Nichols and Hayasaka, 2003b) (and several software libraries have been developed for the purpose (SnPM, 2013; FSL, 2012; Winkler et al., 2014)). Unfortunately, reliable FWER estimates derived via permutation testing come at excessive (and often infeasible) computational cost – often tens of thousands or even millions of permutation samples are required, each of which requires a complete pass over the entire data set. This step alone can run from a few days up to many weeks and even longer (Pantazis et al., 2005; Gaonkar and Davatzikos, 2013a).

Despite the varied advantages of permutation tests, their computational cost has become a critical limiting factor in their usage. To alleviate these runtime costs, ideas that rely on code optimization and parallel computing have been explored (Eklund et al., 2011; A. Eklund, 2012, 2013). These are interesting

strategies but any hardware-based approach will be limited by the amount of resources at hand. Further, as data acquisition technologies continue to get better (leading to higher resolution, larger sized, datasets) in tandem the concurrent slowdown in the predicted increase of processor speeds (Moore's law), it is reasonable to assume that the associated runtime will continue to be a problem in the short to medium term. Clearly, significant gains may be possible if *more efficient schemes* that exploit the underlying structure of the data were available. It seems likely that such algorithms can better exploit the resources (e.g., cloud or compute cluster) one has available as part of a study and may also gain from hardware/code improvements that are being reported in the literature.

Exploiting structure in testing algorithms.

Datasets in many fields of natural sciences, especially biology and medicine, are highly structured. Individual genes and/or individual voxels share a great deal of commonality with other genes and voxels, and it seems reasonable that such correlation can be exploited towards better (or more efficient) statistical algorithms. Even non-localized voxel neighbourhoods in the brain are inherently related because of the underlying biological structures. For example, in genomics, (Cheverud, 2001b) and (Li and Ji, 2005b) used correlations in the data to estimate the effective number of independent tests in a genome sequence to appropriately threshold the test statistics. Also motivated by bioinformatics problems, (Knijnenburg et al., 2009) approached the question of estimating the tail of the distribution of permutation values via an approximation by a generalized Pareto distribution (using fewer permutations). In the context of more general statistical analysis, the authors in (Subramanian et al., 2005) proposed Gene Set Enrichment Analysis (GSEA) which exploits the underlying structure among the genes, to analyze gene-sets (e.g., where sets were obtained from biological pathways) instead of individual genes. If the genes within a gene-set have similar expression pattern, one may see improvements in statistical power.

This idea of exploiting the structure towards efficiency (broadly construed) was more rigorously studied in (Efron and Tibshirani, 2007) and a nice non-parametric Bayesian perspective was offered in (Dahl and Newton, 2007). Within neuroimaging, a similar intuition drives Random Field theory based analysis (Taylor and Worsley, 2008), albeit the objective there is to obtain a less conservative correction, rather than computational efficiency. Recently, motivated

by neuroimaging applications and computational issues, (Gaonkar and Davatzikos, 2013b) derived an analytical approximation of statistical significance maps to reduce the computational burden imposed by permutation tests commonly used to identify which brain regions contribute to a Support Vector Machines (SVM) model. In summary, exploiting the structure of the data to obtain alternative efficient solutions is not new, but we find that in the context of permutation testing on *imaging data*, there is a great deal of voxel-to-voxel correlations that if leveraged properly can, in principle, yield interesting new algorithms.

Main Idea and Contributions

The very same dependencies between voxels, that forced the usage of permutation testing, indicate that the overwhelming majority of work in computing so many highly correlated Null statistics is redundant. Strong dependencies of almost any kind will tend to concentrate most of their co-variation into a low-rank subspace, leaving a high-rank, low-variance residual (Li and Ji, 2005a). In fact, some of the works mentioned earlier (including (Cheverud, 2001a; Li and Ji, 2005a) for genome wide association studies), the rank of some appropriately chosen subspace drives the machinery for computing the effective number of independent tests. The starting point of this work is based on the observation that such a low-rank structure must also appear in permutation test samples. Using ideas from online low-rank matrix completion (He et al., 2012b) we can sample a few of the Null statistics and reconstruct the remainder as long as we properly account for the residual. This allows us to sub-sample at *extremely low rates*, generally $< 1\%$. The object of interest is the permutation testing matrix, $\mathbf{T} \in \mathbb{R}^{v \times L}$ – each row of $\mathbf{T}$ corresponds to one voxel (covariate or feature), and each column is a specific permutation of the labels of the data. Most importantly, the low-rank based modeling of $\mathbf{T}$ does not impose any parametric assumptions on the data itself, and instead is an outcome of the test statistics themselves (as we will see shortly in the later sections). Lastly, the application of testing on brain images is being used as a running example in the presentation – the ideas presented here nevertheless are applicable for any datasets (and application domains). We now list down the contributions of this work. The proposed ideas are summarized in Hinrichs et al. (2013) and Gutierrez-Barragan et al. (2017).

- **Theoretical guarantees.** The basic premise of this paper is that *permu-*

tation testing in high-dimensions (especially, imaging data) is extremely redundant. We show how we can model **T** as a low-rank plus a low-variance residual. We provide two technical results supporting (and justifying) this claim using ideas from random matrix theory, and demonstrate their practical implications. Our first result justifies the modeling assumption itself and several of the components involved in recovering **T**. The second result shows that the error in the *global* maximum null distribution obtained from the estimate of **T** is quite small. Further, using empirical results we validate the assumptions of the proposed approach on several datasets and test statistics.

- **A novel, fast and robust, multiple-hypothesis testing procedure.** Building upon the theoretical development, we propose a fast and accurate algorithm for permutation testing – referred to as RapidPT. We show that compared to existing state-of-the-art libraries for non-parametric testing, the proposed algorithm achieves state-of-the-art runtime performance by reducing the computation time of existing permutation testing software by $20\times$ or more – at *no* cost of loss in accuracy (i.e., without losing any clinical significance performance). Through extensive evaluations on a variety of real-world neuroimaging datasets (from AD studies), we demonstrate the robustness of RapidPT both to the choices of the algorithm's hyper-parameters, the dataset sizes etc. We further identify regimes where the speedup is higher than three orders of magnitude, and show that RapidPT can leverage serial and parallel computing environments seamlessly.
- **Seamless integration of proposed algorithm into standard libraries.** Given the importance and the wide use of permutation testing in applied statistics (including brain imaging and other studies involving high-dimensional and multimodal data), we provide a Matlab implementation of RapidPT integrated as a plugin within the current development version of Statistical Non-parametric Mapping (SnPM) — a widely used non-parametric testing toolbox in neuroimaging studies. Users can invoke RapidPT directly from within the SnPM's well-established graphical user interface, and benefit from SnPM's familiar pre-processing and post-processing capabilities. We also provide a stand-alone installation, and either libraries are documented appropriately.

Relevant Related Work

Exploiting structure to drive permutation testing has been independently investigated by at least few other authors. Of particular relevance, mainly with respect to neuroimaging studies, is the proposed set of approaches in (Winkler et al., 2016) where the authors discuss the accuracy and runtime gains of six approaches that leverage the problem structure to accelerate testing. The present work shares some of the goals and motivation of (Winkler et al., 2016) – specifically, utilizing the algebraic structure of $\mathbf{T}$ – nevertheless, there are substantial technical differences in the present approach, which we outline further below. First, unlike (Winkler et al., 2016), we directly study permutation testing for images at a more fundamental level and seek to characterize mathematical properties of relabeling (i.e., permutation) procedures operating on high-dimensional imaging data. This is different from assessing whether the underlying operations of classical statistical testing procedures can be reformulated (based on the correlations) to reduce computational burden. Second, by exploiting celebrated technical results in random matrix theory, we provide theoretical guarantees for estimation and recovery of $\mathbf{T}$. Few such results were known. Note that empirically, our machinery avoids a large majority of the operations performed in (Winkler et al., 2016). Third, some speed-up strategies proposed in (Winkler et al., 2016) can be considered as special cases of our proposed algorithm — interestingly, if we were to increase the number 'actual' operations performed by RapidPT (from $\approx 1\%$, suggested by our experiments, to 50%), the computational workload begins approaching what is described in (Winkler et al., 2016). Fourth, our experiments suggest that in regimes where we provide $15\times$ or more speedup over classical permutation testing, the strategy in (Winkler et al., 2016) reports a $4\times$ slow down in runtime.

3.2 Permutation Testing in Neuroimaging

We begin with some background on multiple hypothesis testing via permutation testing procedure. As pointed out earlier in Section 3.1, although the problem of measuring outcome efficacy is relevant to any dataset of interest, we restrict our formulation and presentation to voxel-wise (i.e., high dimensional) brain images.

Recall that matrices and vectors will be denoted by bold upper-case and lower-case letters respectively, and scalars will be represented using non-bold

letters. For a matrix $\mathbf{X}$, $\mathbf{X}[i, :]$ denotes the i^{th} row and $\mathbf{X}[i, j]$ denotes the element in i^{th} row and j^{th} column. Randomly sampled permutation testing (Dwass, 1957) is a methodology for drawing samples under the Global (Family-Wise) Null hypothesis (Edgington, 1969b,a; Edgington and Onghena, 2007). Recall that although point-wise test statistics have well characterized univariate Null distributions, the sample maximum usually has no analytic form due to the strong correlations across voxels, as discussed in Section 3.1. Permutation testing is particularly desirable in this setting because it is free of any distribution assumption whatsoever (Nichols and Hayasaka, 2003a).

The basic idea of permutation testing is simple, yet powerful. Suppose we have a set of labeled high dimensional data points, and a univariate test statistic computed at each of the dimensions. Such a statistic can be a t-test which measures the standardized difference of means between the labeled groups. If we randomly permute the labels and recalculate each test statistic, then by construction we get a sample from the Global Null distribution. The maximum over all of these statistics for every permutation sample is then used to construct a histogram, which therefore is a non-parametric estimate of the distribution of the sample maximum of Null statistics. Indeed, when the test corresponds to group differences between samples based on a stratification variable, under the null hypothesis $\mathcal{H}_0$, the grouping labels given to the samples are artificial, i.e., they correspond to the set of all possible *relabellings* of the samples. Within this setting, for a test statistic derived from the real labels, the family-wise error rate (FWER) corrected p-value is then equal to the fraction of permutation samples which were *more extreme*. Note that all of the permutation samples can be assembled into a matrix $\mathbf{T} \in \mathbf{R}^{v \times L}$ where v is the number of comparisons (voxels for images), and L is the number of permutation samples or relabelings. The histogram of all L maximum statistics i.e., the maximum across all samples for a given permutation, is the empirical estimate of the *exact* maximum null distribution under $\mathcal{H}_0$.

In neuroimaging studies, the groups typically correspond to the presence or absence of an underlying disease condition (e.g., CN and AD), or in the more relevant RCT setting the groups may be placebo and intervention. Consider the data to be brain scans, for example MRI, accumulated for the two groups of interest. Whenever $\mathcal{H}_0$ is true, the data sample representing a healthy subject is, roughly speaking, *similar* to a diseased subject. Hence, in principle, *interchanging* the labels of two instances – one from each group – will have no effect on the distribution of the resulting voxel-wise statistics across all the dimensions

(or covariates or features). Hence, under $\mathcal{H}_0$, the recomputed sets of voxel-wise statistics correspond to the same global null distribution, and so we could simply generate one of the two groups given the other. For example, in the RCT setting, no new information – positive (successful drug/intervention) or otherwise – can be inferred from the treatment group. On the other hand, we can use the estimate of global maximum null distribution and test whether some *given* labeling is significant or not. This is done by simply computing the fraction of the max null that is more extreme than the statistic computed at a given voxel with this given labeling., thereby generating resampling risk images for voxel-wise studies (we discuss this in more detail in Section 3.8).

The case for strong null. Observe that when testing *multiple* sets of hypotheses there are two different types of control for the Type 1 error rate (FWER): weak and strong control (Y. Hochberg, 1987). A test is referred to as weak control for FWER whenever the Type 1 error rate is controlled only when all the hypotheses involved are true. In the voxel-wise neuroimaging scenario, this corresponds to $\mathcal{H}_0$ being true, i.e., the two groups differ for *all* the voxels. On the other hand, a test provides strong control for FWER whenever Type 1 error rate is controlled under any combination/proportion of the true hypotheses. It is known that the procedure described above (i.e., using the max null distribution calculated across all voxel-wise statistics) provides strong control of FWER (Holmes et al., 1996). This is easy to verify because the maximum of all voxel-wise statistics is used to compute the corrected p-value, and so, the exact proportion of which hypotheses are true does not matter. Further, testing based on strong control will classify non-activated voxels as activated with a probability upper bounded by the specified significance level α. Hence, it has localizing power (Holmes et al., 1996) – a desirable property in medical studies in particular where covariate/feature significance with respect to some underlying condition is of vital interest. We will focus on such strong control from here on.

NaivePT: The exhaustive Permutation Testing procedure

Figure 3.1 illustrates the classical permutation testing procedure. We refer to this as *naive testing* – NaivePT. Given the data instances from two groups, $\mathbf{X}^1 \in \mathbb{R}^{v \times n_1}$ and $\mathbf{X}^2 \in \mathbb{R}^{v \times n_2}$, where n_1 and n_2 denote the number of subjects/instances from each group respectively. v denotes the number of features,

as in voxels in the brain image. Let $n = n_1 + n_2$ and $\mathbf{X} = [\mathbf{X}^1; \mathbf{X}^2]$ denote the (row-wise) stacked data matrix ($\mathbf{X} \in \mathbb{R}^{v \times n}$). Recall that a permutation of the columns of $\mathbf{X}$ corresponds to a group relabelling. The v distinct voxel-wise statistics are then computed for L such permutations, and used to construct the permutation testing matrix $\mathbf{T} \in \mathbb{R}^{v \times L}$. The empirical estimate of the max null $\mathbf{h}^L$ is the histogram of the maximum of each of the columns of $\mathbf{T}$. This NaivePT, which is also referred to as Monte Carlo permutation tests because of the random sampling involved in generating the statistics, is our baseline.

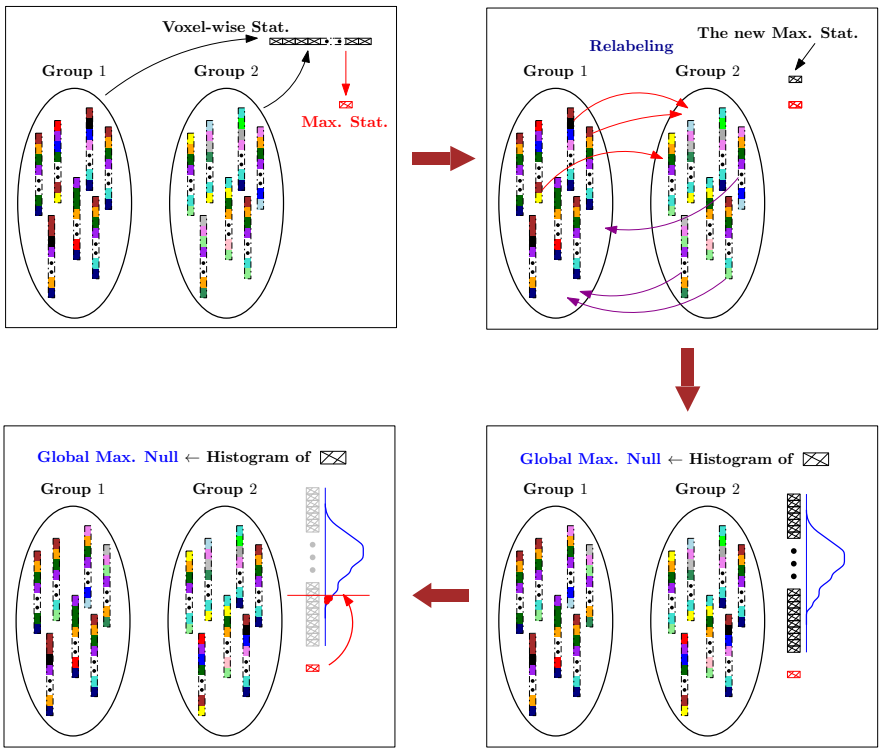


Figure 3.1: NaivePT: Classical permutation testing procedure.

The situation when L is large

Despite the exact unbiased estimation of the null and non-parametric nature of permutation testing, it involves a significant computational footprint, making the exhaustive procedure from Figure 3.1 is very expensive even for reasonably sized studies (retrospective studies or RCTs). First observe that as L increases

the procedure becomes extremely expensive. There are at least a few different reasons for using very large values of L, both from the perspective of sampling efficiency and the need to extract weak signals from the datasets. We discuss these reasons in some detail next.

Sampling artifact.

1. Random sampling methods draw many samples from near the mode(s) of the distribution of interest, but fewer samples from the tail. To characterize the threshold for a small portion of the tail of a distribution, we must invariably draw a very large number of samples just so that the estimate converges. For instance, even in the "most restricted" case where all the permuted maximum statistics are uniform, generating a thousand samples in the $\alpha = 0.01$ tail would require at least *a hundred thousand permutation samples*. In the general setting, the maximum statistics would instead be *skewed* towards some unknown mean, and we would require many more such permutations – a computationally expensive procedure.
2. The expenses in studies involving high-dimensional datasets are further higher because of the sheer number of covariates/features that need to be tested for. Clearly neuroimaging is one such application domain.
3. Lastly, even for reasonable sized datasets (with n on the order of a hundred), the number of permutations L needed to cover even a fraction of all the possible relabellings would be on the order of tens of thousands.

Sensitivity.

1. Strong null control for FWER allows us to localize the significantly different voxels between the two groups. This is done by estimating the threshold corresponds to a given Type 1 error α, and estimating the fraction of voxels whose real labelling statistic falls above this threshold. The fidelity of this threshold depends on the number of observations available to construct the histogram of null distribution, which then directly asks for L to be as high as possible.
2. Ideally we want to obtain a precise threshold for any α, in particular for small α. However, the smallest possible p-value that can be obtained from the empirical null is $\frac{1}{L}$. Therefore, to calculate very low p-values (essential in many applications), L must be very large.

Handling weak signals. A typical characteristic of brain imaging disorders, for instance in the early preclinical stage of AD and other forms of dementia, is that the disease signature is subtle. For instance, recall the discussion from Section 2.2 about the deposition of Amyloid load estimated via PET or atrophy captured in longitudinal MRI in the asymptomatic stage of the disease. These signals are very weak in the early stages of the disease, and even in the RCT setting, the resulting outcomes should be able to pick up such weak differences between treatment and placebo. Pooling from multiple studies – thereby increasing n, is one typical strategy to boost the signal. Any such pooling further entails high confidence tail estimation, which then implies the need for sampling as many permutations as possible.

3.3 A Convex Formulation of Permutation Testing

It turns out that the computational burden of the procedure can be mitigated by exploiting the structural properties of the permutation testing matrix $\mathbf{T}$. Our strategy uses ideas from low-rank matrix completion (LRMC), subspace tracking, and random matrix theory, to exploit the correlated structure of $\mathbf{T}$ and model it in an alternative form. We first briefly discuss matrix completion, followed by an overview of the eigen-spectrum properties of $\mathbf{T}$, which then leads to our proposed model.

Low-Rank Matrix Completion

Given only a small fraction of the entries in a matrix, the problem of low-rank matrix completion (LRMC) (Candès and Tao, 2010) seeks to *recover* the missing entries of the entire matrix. Clearly, with no assumption on the properties of the matrix, such a recovery is ill-posed. Instead, if we *assume* that the column space of the matrix is low-rank and the observed entries are randomly sampled, then the authors of (Candès and Recht, 2009; Candès and Tao, 2010; Jain and Netrapalli, 2015) have shown that, with sufficiently small number of entries, one can recover the orthogonal basis of the row space as well as the expansion coefficients for each column — that is, fully recover the missing entries of the matrix. Specifically, the number of entries required is roughly $r\,\mathrm{poly}(\log(d))$ where r is the rank of the column space and d is the ambient dimension. By placing an ℓ_1-norm penalty on the eigenvalues of the recovered matrix, i.e., the nuclear norm (Fazel et al., 2004; Recht et al., 2010), one optimizes a convex

relaxation of a non-convex objective function which explicitly minimizes the rank. The LRMC problem has received a great deal of attention in the period after the Netflix Prize (Bennett and Lanning, 2007), and numerous applications in machine learning and computer vision have been investigated (Shiratori et al., 2006; Ji et al., 2010; Xu et al., 2013). For details regarding existing algorithms and their analyses including strong recovery guarantees, we refer the reader to (Candès and Recht, 2009; Candès and Tao, 2010; Recht, 2011; Jain and Netrapalli, 2015).

LRMC Formulation. Let us consider a matrix $\mathbf{T} \in \mathbb{R}^{v \times L}$. Denote the set of randomly subsampled entries of this matrix as Ω. This means that we have access to $\mathbf{T}_\Omega$, and our recovery task corresponds to estimating $\mathbf{T}_{\Omega^c}$, where Ω^C corresponds to the complement of the set Ω. Let us denote the estimate of the complete matrix be $\hat{\mathbf{T}}$. The completion problem can be written as the following optimization task,

$$\min_{\hat{\mathbf{T}}} \quad \|\mathbf{T}_\Omega - \hat{\mathbf{T}}_\Omega\|_{\text{Frob}}^2 \quad \text{s.t.} \quad \hat{\mathbf{T}} = \mathbf{UW} \quad \text{and} \quad \mathbf{U} \text{ is orthogonal} \tag{3.1}$$

where $\mathbf{U} \in \mathbb{R}^{v \times r}$ is the low-rank basis of $\mathbf{T}$, i.e., the columns of $\mathbf{U}$ correspond to the orthogonal basis vectors of the column space of $\mathbf{T}$. Here, Ω gives the measured entries and $\mathbf{W}$ is the matrix of coefficients that lets us reconstruct $\hat{\mathbf{T}}$. Although there are several algorithmic procedures for attacking LRMC, we restrict ourselves to the subspace updating procedure proposed in (Balzano et al., 2010; He et al., 2012a), where, given an estimate of the underlying rank r ahead of time, one can compute the orthogonal basis that generates the matrix via gradient descent along an appropriately chosen Grassmannian manifold.

Low-rank plus a long tail in T

Most datasets encountered in the real world (and especially in medical and biological studies) have a dominant low-rank component. While the data may not be *exactly* characterized by a low-rank basis, the residual will not significantly alter the eigen-spectrum of the sample covariance in general. Strong correlations nearly always imply a skewed eigen-spectrum, because as the eigen-spectrum becomes flat, the resulting covariance matrix tends to become sparser – the "uncertainty principle" between low-rank and sparse matrices (Chandrasekaran et al., 2011). Low-rank structure in the data is encountered even more frequently in neuroimaging — unlike natural images in computer vision, there is much

stronger voxel-to-voxel homogeneity in a brain image. Note that efficient representations computed on vision data nevertheless adheres to the low-rank structure, for example, the covariance structure of hidden representations from typical deep networks on vision datasets is observed to have low-dimensional embedding. Getting back to the setting of medical imaging, while performing statistical hypothesis testing on these images, the low-rank structure described above carries through with some modifications.

For purely linear statistics such as sample means, mean differences and so on, the matrix permutation testing matrix $\mathbf{T}$ has the same spectrum as the data. However, non-linearities in the test statistic, e.g., normalizing by pooled variances in t-statistic, will perturb the eigen-spectrum of the original data, contributing a long tail of eigenvalues (see Figure 3.2). This large number of significant singular values needs to be accounted for, if one intends to model $\mathbf{T}$ using low-rank structure. Ideally, we require that this long tail should either decay rapidly, or that it does not overlap with the dominant eigenvalues. This is equivalent to asking that the resulting non-linearities do not *decorrelate* the test statistics, to the point that the matrix $\mathbf{T}$ cannot be approximated by a low-rank matrix with high fidelity. For t-statistics, the non-linearities come from normalization by pooled variances, see for example a two-sample t-test shown in (3.2). Here (μ_1, σ_1) and (μ_2, σ_2) are the mean and standard deviations for the two groups respectively. Since the pooled variances calculated from correlated data $\mathbf{X}$ are unlikely to change very much from one permutation sample to another (except outliers), we expect that the spectrum of $\mathbf{T}$ will resemble that of the data (or sample) covariance, *with the addition of a long, exponentially decaying tail*. More generally, if the non-linearity does not decorrelate the test statistics too much, it will almost certainly preserve the low-rank structure. Although we use the two sample t-test as a running example for the remaining chapter, these observations are true for any test statistic.

$$t = \frac{\mu_1 - \mu_2}{\sqrt{\frac{\sigma_1}{n_1} + \frac{\sigma_2}{n_2}}} \tag{3.2}$$

Does low-rank plus a long tail assumption hold for any medical imaging dataset? The underlying hypothesis of our proposed framework is that $\mathbf{T}$, in general, has this low-rank plus long tail structure. We have tested this on a variety of imaging modalities including those commonly used in medical studies – Arterial Spin Labeling (ASL), Magnetic Resonance Imaging (MRI), two types of

Positron Emission Tomography (PET) modalities including fluorodeoxyglucose (FDG) and Pittsburgh compound B (PiB). we show evidence for the modeling assumption on FDG PET images in Figure 3.2 (trends for the rest of modalities are the same). The low-rank (left column) and the remaining long tail (right column) is clearly seen in these spectrum plots which suggests that the core modeling assumptions are satisfied. The **T** for these plots is constructed using 250 ASL images from two groups (healthy and AD). We note that most of our evaluations in section 3.7 use MRI data. The underlying construct of such high correlations across multiple covariates or predictors is common to most biological datasets beyond brain scans, like genetic datasets including single nucleotide polymorphisms (SNPs) etc.

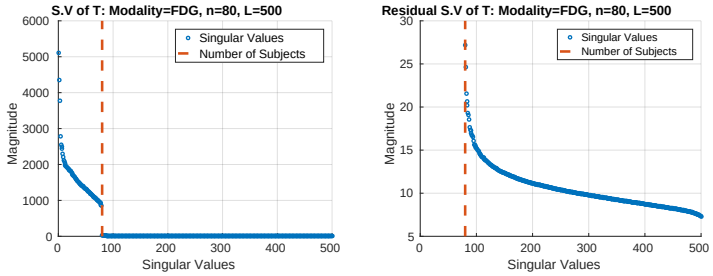


Figure 3.2: Singular value spectra for permutation testing matrix computed from FDG brain images of healthy and diseased subjects.

Overview of Proposed method

If the low-rank structure dominates the long tail described above, then its contribution to **T** can be modeled as a low variance Gaussian i.i.d.residual. A Central Limit argument appeals to the number of independent eigenfunctions that contribute to this residual, and, the orthogonality of eigenfunctions implies that as more of them meaningfully contribute to each entry in the residual, the more independent those entries become. In other words, if this long tail begins at a low magnitude and decays slowly, then we can treat it as a Gaussian i.i.d.residual; and if it decays rapidly, then the residual will perhaps be less Gaussian, but also more negligible. Thus, our algorithm makes no direct assumption about these eigenvalues themselves, but rather that the residual corresponds to a low-variance i.i.d.Gaussian random matrix – its contribution to the covariance of test statistics will be *Wishart* distributed, and from this

property, we can characterize its eigenvalues.

Why should we expect runtime improvements? The low-rank + long tail structure of the permutation testing matrix translates to the following identity, where $\mathbf{S}$ is the *unknown* i.i.d.Gaussian random matrix.

$$\mathbf{T} = \mathbf{UW} + \mathbf{S} \tag{3.3}$$

We do not restrict ourselves to one-sided tests here, and so, $\mathbf{S}$ is modeled to be zero-mean. Later in Section 3.4, we show that this apparent zero-mean assumption is addressed because of a post-processing step. The low-rank portion of $\mathbf{T}$ can be reconstructed by sub-sampling the matrix at Ω using the LRMC optimization from (3.1). Recall from the discussion in Section 3.3 that Ω corresponds to a subset of indices of the entries in $\mathbf{T}$. Instead of computing all voxel-wise statistics for a given relabeling i.e., a column of $\mathbf{T}$, only a small fraction η are computed. This η is referred to as the *sub-sampling rate*. Later in Sections 3.6 and 3.7, we will show that η is very small – approximately on the order of $< 1\%$. Hence, the overall number of entries in Ω — *the number of statistics actually calculated to recover* $\mathbf{T}$ – is $\eta\nu\mathrm{L}$ with $\eta \ll 1$ as opposed to $\nu\mathrm{L}$ (where $\eta = 1$ simply corresponds to NaivePT).

Since the core of the proposal is to model $\mathbf{T}$ by accessing only a small subset of its entries Ω, we refer to it as *rapid permutation testing* – RapidPT. Observe that a significant contributor to the running time of online subspace tracking algorithms, including the LRMC optimization from (3.1), is the module that updates the basis set $\mathbf{U}$; If, on the other hand, a high-fidelity estimate for $\mathbf{U}$ has been found, this re-estimation is unnecessary. Second, the eventual goal of the testing procedure is to recover the global max null as discussed earlier in Section 3.2, which then implies that the residual $\mathbf{S}$ should also be recovered with high fidelity. This directly comes from the fact that sampling procedures tend to underestimate tails of distributions, and hence, an under-estimation of $\mathbf{S}$ results in under-estimation of the max statistic for each permutation (column of $\mathbf{T}$) eventually leading to under-estimation of the histogram of max statistics i.e., the empirical max null. Clearly exact recovery of $\mathbf{S}$ is not possible with $\eta < 1$. Although, for our purposes, we only need its effect on the *distribution of the maximum* per permutation test. A good estimate of the mean and variance of $\mathbf{S}$ – which is being modeled as a random matrix – would then suffice for the max null recovery. We therefore divide the RapidPT algorithm into two steps:

training, and *recovery* which is described in detail in the next section.

3.4 Rapid Permutation Testing – RapidPT

We discuss the specifics of the training and recovery stages of RapidPT, and then present the complete algorithm, followed by some theoretical guarantees regarding consistency and recovery of $\mathbf{T}$. Figure 3.3 summarizes the RapidPT procedure from Algorithm 1.

The training phase

The goal of the training phase is to estimate the basis $\mathbf{U}$. Here, we perform a small number of fully sampled permutation tests, i.e., for approximately a few hundred of the columns of $\mathbf{T}$ (each of which corresponds to a permutation) denoted by ℓ, all the v voxel-wise statistics are computed. This $v \times \ell$ sub-matrix of $\mathbf{T}$ is referred to as the *training set*, denoted by $\mathbf{T}_{ex}$. In our experiments, ℓ was selected to be either a fraction or a multiple of the total number of subjects n as described in section 3.6. From $\mathbf{T}_{ex}$, we estimate the basis $\mathbf{U}$ using sub-sampled matrix completion methods (Balzano et al., 2010; He et al., 2012a), making *multiple* passes over the training set with the (given) sub-sampling rate η, until convergence. This corresponds to initializing $\mathbf{U}$ as a random orthogonal matrix of a pre-determined rank r, and using the columns of $\mathbf{T}_{ex}$ repeatedly to iteratively update it until convergence (see (Balzano et al., 2010; He et al., 2012a) for details regarding subspace tracking).

Once $\mathbf{U}$ is estimated, $\mathbf{W}_{ex}$ is obtained by running a simple least-squares procedure on $\mathbf{T}_{ex}$ and $\mathbf{U}$. The histogram of the entries in $\mathbf{T}_{ex} - \mathbf{U}\mathbf{W}_{ex}$ will then be an estimate of the empirical distribution of the residual $\mathbf{S}$ over the training set. We denote the standard deviation of these *left over* entries as σ – recall that the mean is by definition zero whenever the rank r is sufficiently large. We now discuss a few relevant aspects of this training phase. Notice that in principle, one can estimate $\mathbf{U}$ directly from $\mathbf{T}_{ex}$ by simply computing the leading r principal components. This involves a brute-force approximation of $\mathbf{U}$ by computing the singular-value decomposition of a dense $v \times \ell$ matrix. Even for reasonably small v, this is a costly operation. Second, $\hat{\mathbf{T}}$, by definition, contains a non-trivial residual. We have no direct control on the structure of $\mathbf{S}$ except that it is i.i.d. Gaussian. Clearly, the variance of entries of $\mathbf{S}$ will depend on the fidelity of the approximation provided by $\mathbf{U}$. Since the sub-sampling rate

η (the size of the set Ω compared to νL) is known ahead of time, estimating $\mathbf{U}$ via a subspace-tracking procedure using η fraction of the entries of $\mathbf{T}_{ex}$ (where each column of $\mathbf{T}_{ex}$ modifies an existing estimate of $\mathbf{U}$, one-by-one, without requiring to store all the entries of $\mathbf{T}_{ex}$) directly provides an estimate of $\mathbf{S}$.

Bias-Variance Trade-off. When using a very sparse sub-sampling method i.e., sampling with small η, there is a issue in in estimating $\mathbf{S}$. Clearly, if we use the entire matrix $\mathbf{T}$ to estimate $\mathbf{U}$, $\mathbf{W}$ and $\mathbf{S}$, we will obtain reliable estimates of $\mathbf{S}$. In the $\eta < 1$ setting there is problem arising from sampling: the least-squares objective used in fitting $\mathbf{W}$ (in getting a good estimate of the max null) to such a small sample of entries is likely to grossly underestimate the variance of $\mathbf{S}$ compared to when we use the entire matrix; i.e., the sub-sampling problem is not nearly as over-constrained as it is for the full matrix. This sampling artifact reduces the apparent variance of $\mathbf{S}$, and induces a bias in the distribution of the sample maximum, because extreme values are found less frequently – put it another way, extreme values are sampled less frequently. This artifact has the direct effect of "shifting" the distribution of the sample maximum towards zero. We refer to this as a bias-variance trade-off because, we can think of the need for shift as an outcome of the sub-optimality of the estimate of σ. Hence, the deviation of the true max null from the estimated max null, i.e., *the bias*, is not being captured completely by the estimate of residual variance. As η increases, the apparent sub-optimality reduces, thereby resulting in a better estimate of sample variance. In other words, a good variance estimate results in less bias, and vice versa. We correct for this apparent bias by estimating the amount of the shift during the training phase, and then shifting the recovered sample max distribution by this estimated amount. This empirical shift is denoted by μ, which concludes the training phase.

The recovery phase

In the recovery phase, we sub-sample a small fraction of the entries of each column of $\mathbf{T}$ successively, i.e., for each new relabeling, the voxel-wise statistics are computed over a small fraction of all the voxels to populate $\mathbf{T}_\Omega$. Using this $\mathbf{T}_\Omega$, and the pre-estimated $\mathbf{U}$, the reconstruction coefficients for this column $\mathbf{w} \in \mathbb{R}^{r\times 1}$ are computed. After adding the random residuals to this $\mathbf{Uw}$ – i.i.d. Gaussian with mean μ and standard deviation σ, we have our estimate $\hat{\mathbf{T}}$ for this specific relabeling/permutation. Recall that $\mathbf{S}$ was originally modeled to

be zero-mean (see (3.3)), but the presence of the shift μ suggests a $\mathcal{N}(\mu, \sigma^2)$ distribution instead. Overall, this entails recovering a total of ν voxel-wise statistics from $\eta\nu$ of such entries where $\eta \ll 1$. This process repeats for all the remaining $L - \ell$ columns of $\mathbf{T}$, eventually providing $\hat{\mathbf{T}}$. Once $\hat{\mathbf{T}}$ has been estimated, we proceed exactly as in the NaivePT, to compute the max null and test for the significance of the true labeling.

RapidPT Algorithm

Algorithm 1 and Figure 3.3 summarize RapidPT. The algorithm takes in the input data $\mathbf{X}$, the rank of the basis r, the sub-sampling rate η, the number of training columns ℓ and the total number of columns L as inputs. It returns the estimated permutation testing matrix $\mathbf{T}$ (if needed) and the max null distribution $\mathbf{h}^L$. As described above, the algorithm proceeds by computing $\mathbf{U}$, σ and the shift μ, followed by the $\mathbf{W}$ and $\mathbf{S}$ for the L number of permutations.

3.5 Analysis

We now discuss two results which show that as long as the variance of the residual is below a certain level, we can recover the distribution of the sample maximum. Recall from (3.3) that for low-rank matrix completion methods to be applied we must assume that the permutation matrix $\mathbf{T}$ can be decomposed into a low-rank component $\mathbf{UW}$ plus a high-rank residual matrix $\mathbf{S}$: $\mathbf{T} = \mathbf{UW} + \mathbf{S}$, where $\mathbf{U}$ is a $\nu \times r$ orthogonal matrix that spans the $r \ll \min(\nu, t)$-dimensional column subspace of $\mathbf{T}$, and $\mathbf{W}$ is the corresponding coefficient matrix. We can then treat the residual $\mathbf{S}$ as a random matrix whose entries are i.i.d. zero-mean Gaussian with variance σ^2. We arrive at our first result by analyzing how the low-rank portion of $\mathbf{T}$'s singular spectrum interlaces with the contribution coming from the residual by treating $\mathbf{T}$ as a low-rank perturbation of a random matrix. If this low-rank perturbation is sufficient to dominate the eigenvalues of the random matrix, then $\mathbf{T}$ can be recovered with high fidelity at a low sampling rate (Balzano et al., 2007; He et al., 2012b). Consequently, we can estimate the distribution of the maximum as well, as shown by our second result.

The following development relies on the observation that the eigenvalues of $\mathbf{TT}^\top$ are the squared singular values of $\mathbf{T}$. Thus, rather than analyzing the singular value spectrum of $\mathbf{T}$ directly, we can analyze the eigenvalues of $\mathbf{TT}^\top$ using a result from (Benaych-Georges and Nadakuditi, 2011a). This is important

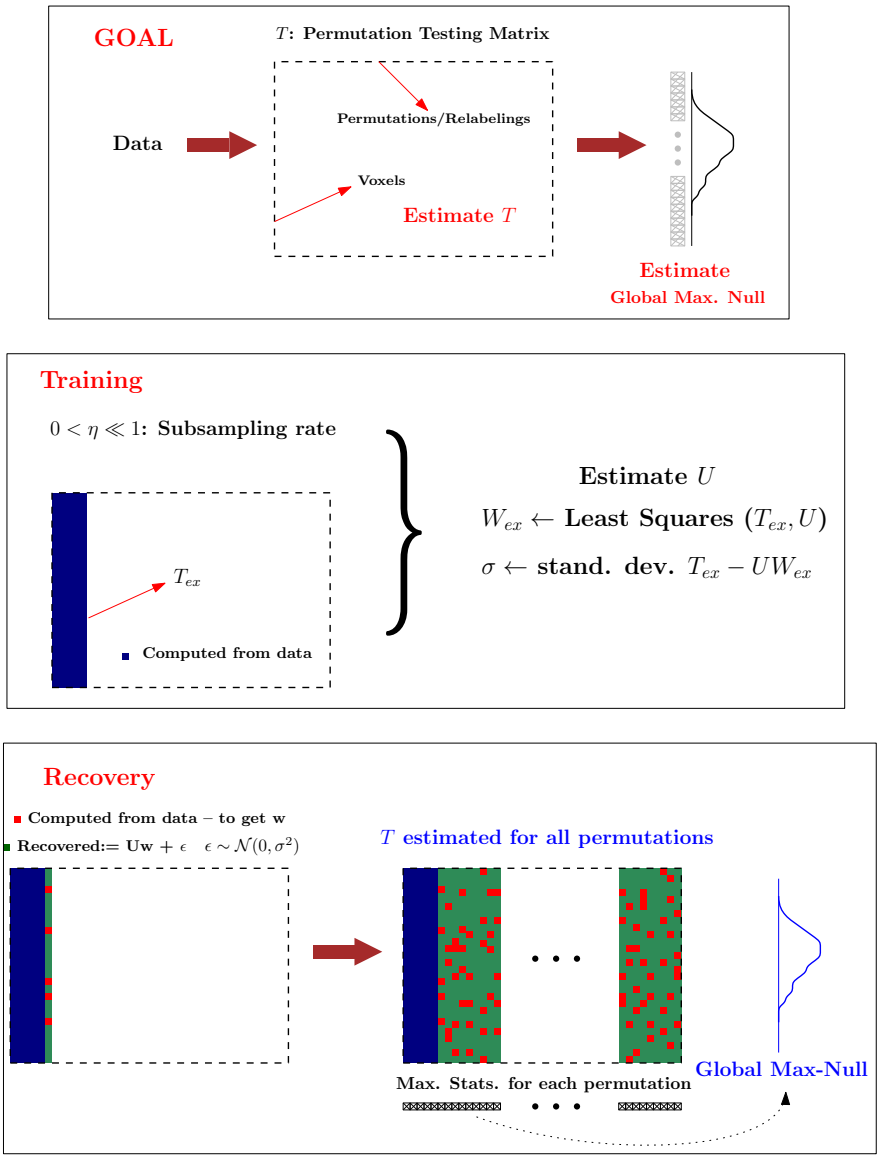


Figure 3.3: The training and recovery steps in RapidPT.

because in order to ensure recovery of **T**, we require that its singular value spectrum will approximately retain the shape of **UW**'s. More precisely, we require that for some $0 < \delta < 1$,

Algorithm 1 The RapidPT algorithm for permutation testing.

Input: $\mathbf{X}^1$, X^2, r, η, L, ℓ, stat
Output: $\hat{\mathbf{T}}$, $\mathbf{h}^L$

TRAINING
$\mathbf{X} = [\mathbf{X}^1; \mathbf{X}^2]$, $n = n_1 + n_2$
$\mathbf{U} \leftarrow$ Rand. Orth., $\mathbf{W}_{ex} = [\varnothing]$
for $i \in 1, \ldots, \ell$ **do**
  $j_1 \ldots, j_n \sim$ Permute$[1, n]$, $\tilde{\mathbf{X}}^1 \leftarrow \mathbf{X}[:, j_1, \ldots, j_{n_1}]$, $\tilde{\mathbf{X}}^2 \leftarrow \mathbf{X}[:, j_{n_1+1}, \ldots, j_n]$
  $\mathbf{T}_{ex}[:, i] \leftarrow \text{test}(\tilde{\mathbf{X}}^1, \tilde{\mathbf{X}}^2)$
  $k_1, \ldots, k_{\lceil \eta\nu \rceil} \sim$ UNIF$[1, \nu]$, $\tilde{\mathbf{T}} \leftarrow \mathbf{T}_{ex}[k_1, \ldots, k_{\lceil \eta\nu \rceil}, i]$
  $\mathbf{U}, \mathbf{W}_{ex}[:, i] \leftarrow$ Subspace-Tracking(r)
end for
$\sigma \leftarrow$ Standard Deviation$\{\mathbf{T}_{ex} - \mathbf{UW}_{ex}\}_\Omega$, $\mu \leftarrow \sup_i$ Max$\{\mathbf{T}_{ex}[:, i] - \mathbf{UW}_{ex}[:, i]\}$
for $i \in 1, \ldots, \ell$ **do**
  $\hat{\mathbf{T}}[:, i] \leftarrow \mathbf{T}[:, i]$
end for

RECOVERY
for $i \in \ell + 1, \ldots, L$ **do**
  $k_1, \ldots, k_{\lceil \eta\nu \rceil} \sim$ UNIF$[1, \nu]$, $j_1 \ldots, j_n \sim$ Permute$[1, n]$
  $\tilde{\mathbf{X}}^1 \leftarrow \mathbf{X}[k_1, \ldots, k_{\lceil \eta\nu \rceil}, j_1, \ldots, j_{n_1}]$, $\tilde{\mathbf{X}}^2 \leftarrow \mathbf{X}[k_1, \ldots, k_{\lceil \eta\nu \rceil}, j_{n_1+1}, \ldots, j_n]$
  $\tilde{\mathbf{T}} \leftarrow \text{test}(\tilde{\mathbf{X}}^1, \tilde{\mathbf{X}}^2)$, $\mathbf{W}[:, i] \leftarrow$ Complete$(\mathbf{U}, \tilde{\mathbf{T}}, k_1, \ldots, k_{\lceil \eta\nu \rceil})$
  $\mathbf{s} \leftarrow$ i.i.d$\mathcal{N}^\nu(0, \sigma^2)$, $\hat{\mathbf{T}}[:, i] \leftarrow \mathbf{UW}[:, i] + \mathbf{s}$
end for
for $i \in 1, \ldots, L$ **do**
  if $i \leqslant \ell$ **then**
    $m_i \leftarrow$ Max$(\hat{\mathbf{T}}[:, i])$
  else
    $m_i \leftarrow$ Max$(\hat{\mathbf{T}}[:, i]) + \mu$
  end if
end for
$\mathbf{h}^L \leftarrow$ Histogram$(m_1, \ldots, m_L)$

$$|\tilde{\phi_i} - \phi_i| < \delta\phi_i \qquad i = 1, \ldots, r; \qquad \tilde{\phi_i} < \delta\phi_r \qquad i = r+1, \ldots, \nu \tag{3.4}$$

where ϕ_i and $\tilde{\phi_i}$ are the singular values of $\mathbf{UW}$ and $\mathbf{P}$ respectively. Recall that in this analysis $\mathbf{T}$ is the perturbation of $\mathbf{UW}$. Theorem 3.1 relates the rate at which eigenvalues are perturbed, δ, to the parameterization of $\mathbf{S}$ in terms of σ^2. The theorem's principal assumption also relates σ^2 inversely with the number of columns of $\mathbf{T}$, which is just the number of trials t. Note however that the process may be split up between several matrices $\mathbf{T}_i$, and the results can then be

combined. For purposes of applying this result in practice we may then choose a number of columns t which gives the best bound. Theorem 3.1 also assumes that the number of trials t is greater than the number of voxels v, which is a difficult regime to explore empirically. Thus, our numerical evaluations cover the case where $t < v$, while Theorem 3.1 covers the case where t is larger. From the definition of $\mathbf{T}$ in (3.3), we have,

$$\mathbf{T}\mathbf{T}^T = \mathbf{U}\mathbf{W}\mathbf{W}^T\mathbf{U}^T + \mathbf{S}\mathbf{S}^T + \mathbf{U}\mathbf{W}\mathbf{S}^T + \mathbf{S}\mathbf{W}^T\mathbf{U}^T. \tag{3.5}$$

We first analyze the change in eigenvalue structure of $\mathbf{S}\mathbf{S}^T$ when perturbed by $\mathbf{U}\mathbf{W}\mathbf{W}^T\mathbf{U}^T$, which has r non-zero eigenvalues. The influence of the cross-terms $\mathbf{U}\mathbf{W}\mathbf{S}^T$ and $\mathbf{S}\mathbf{W}^T\mathbf{U}^T$ is addressed later. Thus, we have the following theorem.

Theorem 3.1 (**Perturbation of eigenvalues**). *Denote that r non-zero eigenvalues of $\boldsymbol{Q} = \boldsymbol{U}\boldsymbol{W}\boldsymbol{W}^T\mathbf{U}^T \in \mathbf{R}^{v\times v}$ by $\lambda_1 \geqslant \lambda_2 \geqslant, \ldots, \lambda_r > 0$; and let $\mathbf{S}$ be a $v \times t$ random matrix such that $\mathbf{S}_{i,j} \sim \mathcal{N}(0, \sigma^2)$, with unknown σ^2. As $v, t \to \infty$ such that $\frac{v}{t} \ll 1$, the eigenvalues $\tilde{\lambda_i}$ of the perturbed matrix $\boldsymbol{Q} + \boldsymbol{S}\boldsymbol{S}^T$ will satisfy*

$$|\tilde{\lambda_i} - \lambda_i| < \delta\lambda_i \qquad i = 1, \ldots, r; \qquad \tilde{\lambda_i} < \delta\lambda_r \qquad i = r+1, \ldots, v \tag{3.6}$$

for some $0 < \delta < 1$, whenever $\sigma^2 < \frac{\delta\lambda_r}{t}$

Proof. The first half of the proof emulates Theorem 2.1 from (Benaych-Georges and Nadakuditi, 2011b). Consider the matrix $\mathbf{X} = \sqrt{t}\mathbf{S}$. By the structure of $\mathbf{S}$, each entry of $\mathbf{X}$ is i.i.d.Gaussian with zero–mean and variance $\sigma^2 t$. Let $\mathbf{Y} = \frac{1}{t}\mathbf{X}\mathbf{X}^T$ and denote its ordered eigenvalues as $\gamma_i, i = 1, \ldots, v$ (large to small). Consider the random spectral measure

$$\mu_v(A) = \tfrac{1}{v}\#\{\gamma_i \in A\}, \qquad A \subset \mathbb{R}$$

The Marchenko–Pastur law (Marčenko and Pastur, 1967) states that as $v, t \to \infty$ such that $\frac{v}{t} \leqslant 1$, the random measure $\mu_v \to \mu$, where $d\mu$ is given by

$$d\mu(a) = \tfrac{1}{2\pi\sigma^2 t\gamma a}\sqrt{(\gamma_+ - a)(a - \gamma_-)}\mathbf{1}_{[\gamma_-,\gamma_+]}da$$

where $\gamma = \frac{v}{t}$. Here $\mathbf{1}_{[\gamma_-,\gamma_+]}$ is an indicator function that is non–zero on $[\gamma_-, \gamma_+]$. $\gamma_\pm = \sigma^2 t(1 \pm \sqrt{\gamma})^2$ are the extreme points of the support of μ. It is well known that the extreme eigenvalues converge almost surely to $\gamma_\pm$ (Edelman, 1988). Since $v, t \to \infty$ and $\gamma = \frac{v}{t} \ll 1$, the length of $[\gamma_-, \gamma_+]$ is much smaller than the values in it. Hence we have,

$$\gamma_{\pm} \sim \sigma^2 t(1 \pm 2\sqrt{\gamma}) \quad ; \quad \sqrt{(\gamma_+ - a)(a - \gamma_-)} \ll a$$

and the new $d\mu(a)$ is given by

$$d\mu(a) = \frac{\sqrt{(\sigma^2 t(1+2\sqrt{\gamma}) - a)(a - \sigma^2 t(1-2\sqrt{\gamma}))}}{2\pi\gamma\sigma^4 t^2} \mathbf{1}_{[\sigma^2 t(1-2\sqrt{\gamma}), \sigma^2 t(1+2\sqrt{\gamma})]} da$$

$$= \frac{1}{2\pi\gamma\sigma^4 t^2} \sqrt{4\gamma\sigma^4 t^2 - (a - \sigma^2 t)^2} \mathbf{1}_{[\sigma^2 t(1-2\sqrt{\gamma}), \sigma^2 t(1+2\sqrt{\gamma})]} da$$

The form we have derived for $d\mu(a)$ shares some similarities with $d\mu_X(x)$ in Section 3.1 of (Benaych-Georges and Nadakuditi, 2011b). The analysis in (Benaych-Georges and Nadakuditi, 2011b) takes into account the phase transition of extreme eigen values. This is done by imitating a time–frequency type analysis on compact support of extreme spectral measure i.e. using Cauchy transform. For our case, the Cauchy transform of $\mu(a)$ is

$$G_\mu(z) = \frac{1}{2\gamma\sigma^4 t^2} \left(z - \sigma^2 t - \text{sgn}(z) \sqrt{(z - \sigma^2 t)^2 - 4\gamma\sigma^4 t^2} \right)$$

$$\text{for} \quad z \in (\infty, \sigma^2 t(1-2\sqrt{\gamma})) \cup (\sigma^2 t(1+2\sqrt{\gamma}), \infty)$$

Since we are interested in the asymptotic eigen values (and $\gamma \ll 1$), $G_\mu(\gamma_\pm)$ and the functional inverse $G_\mu^{-1}(\theta)$ are

$$G_\mu(\gamma_+) = \tfrac{1}{\sigma^2 t\sqrt{(\gamma)}} \;\; ; \;\; G_\mu(\gamma_-) = -\tfrac{1}{\sigma^2 t\sqrt{(\gamma)}} \;\; ; \;\; G_\mu^{-1}(\theta) = \sigma^2 t + \tfrac{1}{\theta} + \gamma\sigma^4 t^2 \theta$$

Hence, the asymptotic behavior of the eigen values of perturbed matrix $\mathbf{Q} + \mathbf{S}\mathbf{S}^T$ is (observing that $\mathbf{S}\mathbf{S}^T = \mathbf{Y}$ and $\mathbf{Q}$ has r non–zero positive eigen values)

$$\tilde{\lambda_i}(i = 1, \ldots, r) \;\; \approx \;\; \begin{cases} \lambda_i + \sigma^2 t + \frac{\gamma\sigma^4 t^2}{\lambda_i} & \text{for} \quad \lambda_i > \gamma\sigma^2 t \\ \gamma\sigma^2 t & \text{else} \end{cases} \tag{3.7}$$

$$\tilde{\lambda_i}(i = r+1, \ldots, \nu) \;\; \approx \;\; \sigma^2 t(1 - 2\sqrt{(\gamma)}) \tag{3.8}$$

With $\tilde{\lambda_i}, i = 1, \ldots, \nu$ in hand, we now bound the unknown variance σ^2 such that (3.6) is satisfied. We only have two cases to consider,

(1) $\lambda_i > \gamma\sigma^2 t$, $i = 1, \ldots, r$ (2) $\lambda_i \leqslant \gamma\sigma^2 t$, $i = k, \ldots, r$ (for some $k \geqslant 1$)

We constrain the unknown σ^2 such that case (2) does not arise. Substituting for $\tilde{\lambda_i}$'s from (3.7) in (3.6), we get,

$$\sigma^2 t + \tfrac{\gamma\sigma^4 t^2}{\lambda_i} < \delta\lambda_i \quad ; \quad \lambda_i > \gamma\sigma^2 t \quad ; \quad \sigma^2 t(1 - 2\sqrt{(\gamma)}) < \delta\lambda_r$$

These inequalities will hold when $\sigma^2 < \frac{\delta\lambda_r}{t}$ (since $\gamma =\ll 1$, $\delta < 1$ and $\lambda_1 \geqslant \lambda_2 \geqslant, \ldots, \lambda_r$). □

Note that the missing cross-terms would not change the result of Theorem 3.1 drastically, because $\mathbf{UW}$ has r non-zero singular values and hence $\mathbf{UW}\,\mathbf{S}^T$ is a low-rank projection of a low-variance random matrix, and this will clearly be

dominated by either of the other terms. Having justified the model $\mathbf{T} = \mathbf{UW} + \mathbf{S}$ in (3.3), the following theorem shows that the empirical distribution of the maximum Null statistic approximates the true distribution.

Theorem 3.2 (**Recovery of Max. Null**). *Let* $m_t = \max_i \mathbf{T}_{i,t}$ *be the maximum observed test statistic at permutation trial* t*, and similarly let* $\hat{m}_t = \max_i \hat{\mathbf{T}}_{i,t}$ *be the maximum reconstructed test statistic. Further, let the maximum reconstruction error be* ϵ*, such that* $|\mathbf{T}_{i,t} - \hat{\mathbf{T}}_{i,t}| \leqslant \epsilon$*. Then, for any real number* $k > 0$*, we have,*

$$Pr\left[m_t - \hat{m}_t - (b - \hat{b}) > k\epsilon\right] < \frac{1}{k^2}$$

where b *is the bias term described in Section 2, and* $\hat{b}$ *is its estimate from the training phase.*

Proof. Recall that there is a bias term in estimating the distribution of the maximum which must be corrected for this is because $\text{var}(\hat{S})$ underestimates $\text{var}(S)$ due to the bias/variance tradeoff. Let b be this difference:

$$b = \mathbb{E}_t\left[\max_i \mathbf{T}_{i,l}\right] - \mathbb{E}_t\left[\max_i \hat{\mathbf{T}}_{i,l}\right].$$

Further, recall that we estimate b by taking the difference of mean sample maxima between observed and reconstructed test statistics over the training set, giving $\hat{b}$, which is an unbiased estimator of b — it is unbiased because a difference in sample means is an unbiased estimator of the difference of two expectations.

Let $\delta_l = m_l - \hat{m}_l$. To show the result we must derive a concentration bound on δ_l, which we will do by applying Chebyshev's inequality. In order to do so, we require an expression for the mean and variance of δ_l. First, we derive an expression for the mean. Taking the expectation over l of $m_l - \hat{m}_l$ we have,

$$\begin{aligned}\mathbb{E}_t\left[m_l - \hat{m}_l\right] &= \mathbb{E}_t\left[\max_i \mathbf{T}_{i,l} - \max_i \hat{\mathbf{T}}_{i,l} - \hat{b}\right] \\ &= \mathbb{E}_t\left[\max_i \mathbf{T}_{i,l}\right] - \mathbb{E}_t\left[\max_i \hat{\mathbf{T}}_{i,l}\right] - \hat{b} = b - \hat{b}\end{aligned}$$

where the second equality follows from the linearity of expectation.

Next, we require an expression for the variance of δ_l. Let i be the index at which the maximum observed test statistic occurs for permutation trial l, and likewise let j be the index at which the maximum reconstructed test statistic occurs. Thus we have,

$$\begin{aligned}\mathbf{T}_{i,l} &\leqslant \hat{\mathbf{T}}_{i,l} + \epsilon \;\leqslant \hat{\mathbf{T}}_{j,l} + \epsilon \\ \mathbf{T}_{i,l} &\geqslant \hat{\mathbf{T}}_{j,l} \qquad \geqslant \hat{\mathbf{T}}_{j,l} - \epsilon,\end{aligned}$$

and so we have that $|m_l - \hat{m}_l| < 2\epsilon$, and so $\text{var}(m_l - \hat{m}_l) \leqslant \epsilon^2$. Applying Chebyshev's bound, $\Pr\left[m_l - \hat{m}_l - (b - \hat{b}) > k\epsilon\right] < \frac{1}{k^2}$ which completes the proof. □

3.6 Experimental Setup

We evaluate RapidPT in multiple phases to exhaustively test the algorithm for both performance and speed. We outline these phases here, followed by the hyperparameters choices used for RapidPT and the methods used to quantify accuracy of recovering max Null. Observe that since the presentation of the ideas of RapidPT have used the two sample t-tests as a running example, our choice of statistics here was the same two sample testing – although the algorithm is agnostic to the choice of the test.

Phase I: Our first set of evaluations are simulations. The two goals here are to empirically demonstrate the requirements on the algorithm hyperparameters that will guarantee an accurate recovery of the max null; and to evaluate the performance of RapidPT on multiple synthetic datasets generated by changing the strength of group-wise differences and the sparsity of the signal (e.g., how many voxels are different between groups). These empirical results are compared to the analytical bounds governed by the theory, and one of the outcomes of these simulations is to walk through the choices of hyperparameters and RapidPT's robustness to them.

Phase II: Building upon these simulations, we then evaluate RapidPT against the classical version of the test i.e., NaivePT. For these evaluations, we use four separate neuroimaging datasets of AD, MCI and CN subjects – we simply refer to these as Dataset $A - D$. The demographic characteristic of the subjects are different in the datasets. The first of these is from ADNI study, while the others are part of WRAP study (see the discussion from Section 2.3). Our evaluations focus on three main questions: **(i)** Can we recover an acceptable approximation of the maximum statistic Null distribution from an approximation of the permutation test matrix? **(ii)** What degree of computational speedup can we expect at various sub-sampling rates, and how does this affect the trade-off with

approximation error? **(iii)** How sensitive is the estimated α-level threshold with respect to the recovered Null distribution? In all our experiments, the rank estimate for subspace tracking (to construct the low–rank basis **U**) was taken as the number of subjects.

Phase III: Further building upon these evaluations, we compare RapidPT to the current state-of-the-art Matlab implementation for permutation testing in neuroimaging, Statistical NonParametric Mapping SnPM (Nichols and Holmes, 2002). SnPM implements the permutation test exactly like NaivePT, however, it relies on smart matrix allocation and computation mechanisms (Nichols and Hayasaka, 2003b; Winkler et al., 2016). The questions we are interested include, the exhaustive accuracy and runtime estimates (as listed above) as well as the resampling risk (which we define later in Section 3.6) images. Clearly, for runtime performance, SnPM acts as the default baseline. For these evaluations, we restrict ourselves to the AD and CN groups from ADNI data. The very small differences between the results provided by SnPM and NaivePT offer a secondary reference point that tells us an acceptable range for RapidPT's results.

Data for Phase I: We used two simulated setups. While one fixes the dataset and changes algorithmic hyperparameters (listed below in Section 3.6), the other uses multiple datasets with same hyperparameter setting. The first dataset has $n = 30$ synthetic images composed of $v = 20$k voxels. The signal in each voxel is derived from one of the normal distributions: $\mathcal{N}(\mu = 0, \sigma^2 = 1)$ and $\mathcal{N}(\mu = 1, \sigma^2 = 1)$. Two groups were then constructed with 15 images in each and letting 1% (200 voxels) exhibit voxel-wise group differences. The signal in the remaining 99% of the voxels was assumed to come from $\mathcal{N}(\mu = 0, \sigma^2 = 1)$. The second simulated dataset has 48 synthetically generated datasets, each with $v = 20$k voxels. The datasets were generated by varying: the number of images ($n = 60$, $n = 150$, $n = 600$), the strength of the signal (i.e., deviation of μ in $\mathcal{N}(\mu, 1)$ from $\mathcal{N}(0, 1)$) and the sparsity of the signal (percentage of voxels showing group differences). Each dataset here was split into two *equally sized* groups for testing, The first group in all the datasets was generated from $\mathcal{N}(\mu = 0, \sigma^2 = 1)$. For second group, we chose $\{1\%, 5\%, 10\%, 25\%\}$ of the voxels from one of four distributions – $\mathcal{N}(\mu = 1, \sigma^2 = 1)$, $\mathcal{N}(\mu = 5, \sigma^2 = 1)$, $\mathcal{N}(\mu = 10, \sigma^2 = 1)$, $\mathcal{N}(\mu = 25, \sigma^2 = 1)$). The signal in the remaining voxels in the second group was also obtained from $\mathcal{N}(\mu = 0, \sigma^2 = 1)$.

Data for Phase II & **III**: As mentioned earlier, the later evaluation phases we use imaging data from ADNI and/or WRAP studies, specifically the grey matter

issue maps from MRI data. All the data were preprocessed appropriately (the discussion in Section 2.3 provides these details). The preprocessed data is then accumulated as data matrix $\mathbf{X} \in \mathbb{R}^{n \times v}$ where n is the appropriate number of subjects used, and v is typically on the order of $400 - 600$k voxels. Note that some uninformative voxels may be discarded prior to constructing $\mathbf{X}$ – voxels whose mean signal ≈ 0.

Hyperparameters

As outlined in Algorithm 1, there are three high-level input parameters that will impact the performance of the procedure: number of training samples (ℓ), sub-sampling rate (η) and the number of permutations (L). To demonstrate the robustness of the algorithm, we explored several combinations of these hyperparameters for each dataset. This also helps us identify the general regimes under which RapidPT will be a much superior alternative to regular permutation testing. Recall that the baselines for a given combination of these hyperparameters are given by the max null distribution constructed by SnPM and NaivePT.

Number of Training Samples: The number of training samples, ℓ, determines the number of columns of the permutation testing matrix $\mathbf{T}$ that were used to estimate the basis of the subspace. Recall from Section 3.3 that ℓ also corresponds to the number of passes used to estimate the apparent shift that corrects for the bias-variance tradeoff. We use the total number of subjects n as a guide to pick a sensible ℓ. The rationale here is that $\mathbf{T}$ has at least n leading eigen vectors (since n is the number of subjects). Since it makes sense to use as many columns as possible to estimate the best shift that compensates the bias-variance tradeoff, we expect ℓ to be as close as possible to n. Further, $\ell > n$ is unnecessary because the computational overhead of training phase is much higher than recovery, since subspace tracking runtime depends on the number of columns of $\mathbf{T}_{ex}$ used to estimate the basis. Hence, for simulations, given a dataset size n, the choices of ℓ include $\frac{n}{3}$, n and $2n$. While for the real data experiments, ℓ will be $\frac{n}{2}$, $\frac{3n}{4}$, n and $2n$.

Sub-sampling rate: The sub-sampling rate, η, is the percentage of all the entries of $\mathbf{T}$ and $\mathbf{T}_{ex}$ that we will sample, i.e., actually compute. In other words, $\eta v \ell$ number of computations are necessary for computing the basis (the entire $v\ell$ is nevertheless needed for estimating the residual), while during recovery, within any column (permutation), ηv number of statistics are calculated to recover the

rest. While for simulations rates ranging from 0.5% all the way up to $> 50\%$ were used, the experiments on real data used small sub-sampling rates – from 0.1% up to 5%.

Number of Permutations: The number of permutations, L, determines the total number of columns in **T**. By varying L we are able to see how the size of **T** affects the accuracy of the algorithm and also how it scales compared to a standard permutation testing implementations (NaivePT and SnPM). The range of L used include 5 thousand to > 150 thousand.

Accuracy Benchmarks

We used three different measures – Kullback-Leibler Divergence (KL-Divergence), t-thresholds/p-values and the resampling risk – to assess the accuracy of the recovered max Null distribution by RapidPT. The baseline algorithms as mentioned earlier are NaivePT and SnPM.

- *Kullback-Leibler (KL) Divergence*: The KL-Divergence provides a measure of the statistical distance between two probability distributions. In our setting, one of these distributions corresponds to the max Null coming from SnPM or NaivePT, and the other an estimate obtained from RapidPT. The goal is to measure the circumstances under which RapidPT provides a good estimate of the true max null distribution, and if there are cases where the distances are too large or unsatisfactory.

- t-*Thresholds*/p-*values*: Once we have evaluated whether all methods recover a similar max null distribution, we analyze if t-thresholds associated with a given p-value calculated from each max null distribution are also similar. This is essentially one way to compare the tail mismatch between the ground truth and RapidPT. Minimal tail mismatch is critical because the threshold that determines whether to call a voxel significant or not depends entirely on the proportion of max Null in the tails, and the activations picked-up by SnPM and/or NaivePT should be the same as those detected by RapidPT.

- *Resampling Risk*: Building upon the aspect of tail mismatch, observe that two methods can recover a similar max null distribution and p-values, and yet partially disagree in *which voxels* should be classified as statistically significant (e.g., within a group difference analysis). The resampling risk is a more well-defined way to address the tail mismatch. It is the

probability that the decision of accepting/rejecting the null hypothesis differs between two methods (Jockel, 1984). Let ν_{A} and ν_{B} denote the set of voxels whose null hypotheses were rejected according to the max null derived from say Method A and B respectively. Further, let ν_{AD} denote their set intersection, with $\ominus$ and $|\cdot|$ denoting the set difference and cardinality respectively. The resampling risk can then given by

$$\text{risk} = \frac{1}{2}\left(\frac{\nu_{\mathrm{A}} \ominus \nu_{\mathrm{AB}}}{\nu_{\mathrm{A}}} + \frac{\nu_{\mathrm{B}} \ominus \nu_{\mathrm{AB}}}{\nu_{\mathrm{B}}}\right) \tag{3.9}$$

Implementation Details. All evaluation runs reported in this chapter, specifically the runtime calculations, were performed on machines with the same hardware configuration. Each machine comprised of 16 cores with two Intel(R) Xeon(R) CPU $\mathrm{E5}-2670$. One MATLAB process can then use at most 16 processors. For valid runtime performance evaluations, we forced each process of MATLAB to use a specific number of threads. First, we forced MATLAB to only use a single thread (single core) when running SnPM, RapidPT, and NaivePT. The performance results on such a single threaded environment attempts to emulate a scenario where the application was running on an older laptop/workstation serially. We then allowed MATLAB to use all 16 threads available to demonstrate how RapidPT is also able to leverage a parallel computing environment to reduce its overall runtime. Although all machines had the same hardware setup, to further ensure that we were making a fair runtime performance comparison, all time measurements shown in plots/figures from Section 3.7 were obtained from the same machine.

3.7 Results

Simulation Results: Phase I

Figures 3.4 and 3.5 show the snapshot summary of the KL divergence between the max null recovered by NaivePT and RapidPT on 48 synthetically generated datasets. Observe from Figure 3.4 that once the sub-sampling rate, η, exceeds the minimum value established by LRMC theory, RapidPT is able to accurately recover the max null with a KL divergence < 0.01. Increasing ℓ can lead to slightly lower KL divergence as seen in the right most plot, and increasing L improves the accuracy as well. A through discussion of how to choose the important hyperparameters is in Section 3.8. In Figure 3.5, $\mathrm{L} = 50000$, $\eta = 2\eta_{\mathtt{min}}$,

and $\ell = \mathfrak{n}$, where η_{min} refers to the theoretical minimum sub-sampling rate and $\mathfrak{n}$ is the dataset size. As expected, the strength or sparsity of the signal does not have an impact on the performance of RapidPT. The dataset size, however, does have a slight impact on the accuracy but we still find that the recovered t-thresholds for the smaller datasets are within 2% of the true threshold. Figure 3.6 shows the log percent difference between the t-thresholds for different p-values obtained from max null recovered by NaivePT and RapidPT. Similar to Figure 3.4, it is evident that once the strict requirement on the minimum value of η is achieved, we obtain a reliable t-threshold (differing from true threshold by $< 10^{-3}\%$. Additionally, increasing the number of training samples (progression of plots from left to right) gives a improvement in the accuracy, however, this is not incredibly significant since we are already at negligible errors. Overall we see that RapidPT is able to estimate accurate thresholds even in extremely low p-value regimes.

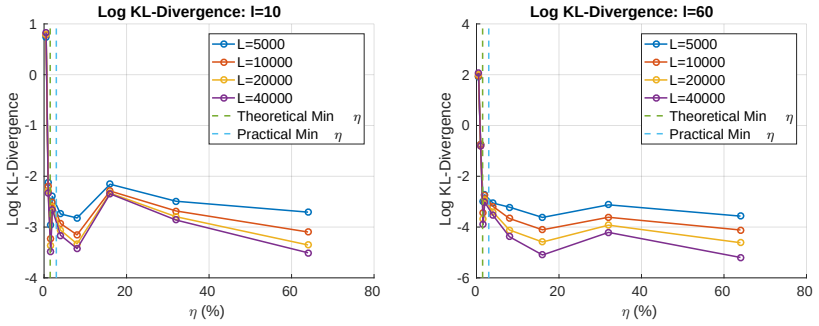


Figure 3.4: KL divergence between true max null and RapidPT.

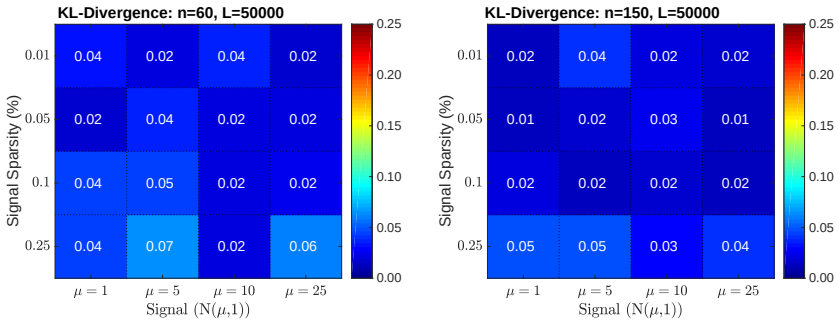


Figure 3.5: KL divergence between true max null RapidPT.

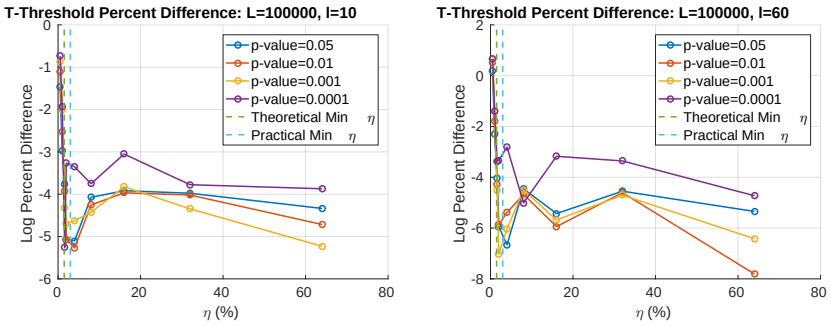


Figure 3.6: Percent difference between the t-thresholds (for different p-values).

Can we recover the Maximum Null?

Building upon the previous simulation results, the first set of results on real datasets suggest that our proposal can recover the max Null, and the recovery comes at much less computation cost. Figure 3.7 summarizes this observation on all the four datasets (with different demographics) used in these Phase II evaluations, and 20 different sub-sampling rates (ranging from 0.1% to 10%). The first observation here is that the KL measures of the recovered Null to the true distribution are $< e^{-5}$ for sampling rates more than 0.4%. In other words, as we saw earlier in simulations, above a certain minimum sub-sampling rate ($\sim 0.3\%$), the KL measures do not change drastically as the rate is increased. Note that we also estimated alternate divergence measures like Bhattacharya distance (BD), and we see that RapidPT recovers both the shape (low BD) and position (low KL) of the null to high accuracy even at extremely low sub-sampling. Since the estimation is prone to noise (for example, random sampling, choice of ℓ etc.), we also computed error bars pertaining to multiple realizations on the different sampling rates – and the bars are significantly smaller (within $< 10\%$ of the KL).

What is the computational speedup?

The speedup is substantial – see Figure 3.7. RapidPT achieved at least 30 times decrease in computation time in the low sampling regime ($< 1\%$). Around $0.5\% - 0.6\%$ sub-sampling (where the KL and BD are already $< e^{-5}$), the computation speed-up factor averaged over all datasets was $45\times$. This shows that RapidPT achieves good accuracy (low KL and BD) in tandem with high compu-

tational speed up, especially, for $0.4\% - 0.7\%$ sampling rates. However as seen from Figure 3.7, there is a trade–off between the speedup factor and approximation error. Overall the highest computational speedup factor achieved at a recovery level of e^{-5} on KL and BD is around $50\times$ (and this occurred around $0.4\% - 0.5\%$ sampling rate, refer to Figure 3.7). It was observed that a speedup factor of upto $55\times$ was obtained for two out of the four datasets used here.

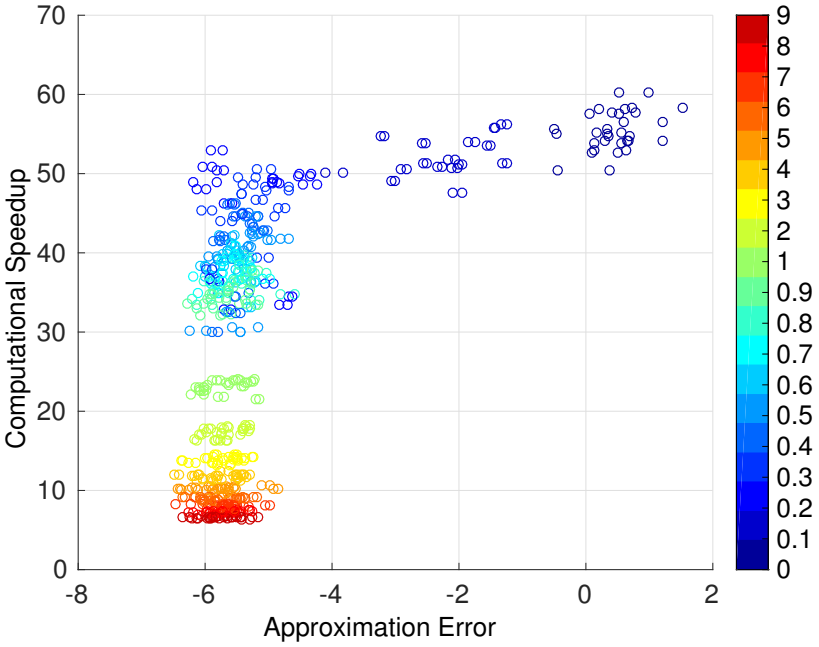


Figure 3.7: Scatter plot of computational speedup vs. approximation error.

How stable is the estimated α-threshold (clinical significance)?

Similar to the simulations from Figure 3.6 and 3.5, our evaluations on real data also suggest that the threshold is stable. Table 3.1 summarizes this clinical significance. The shape and area in the tail of the null distribution are being recovered with high accuracy – which we show via the absolute differences of the estimated thresholds on all the datasets at 4 different α levels in Table 3.1. The errors for $1 - \alpha = 0.95, 0.99$ are at most 0.16. The increase in error for $1 - \alpha > 0.995$ is a sampling artifact and is expected. However, in a few

cases, the error at 0.5% is slightly higher than that at 0.3% suggesting that the recovery is noisy (see Sec. 3.7). We discuss more about this while presenting resampling risk images in Section 3.7. Overall the estimated α-thresholds are both faithful and stable. These results are vital for practical usage of RapidPT, and we discuss more about the thresholds shortly.

Data name	Sampling rate	$1-\alpha$ level			
		0.95	0.99	0.995	0.999
A	0.3%	0.16	0.11	0.14	0.07
	0.5%	0.13	0.08	0.10	0.03
B	0.3%	0.02	0.05	0.03	0.13
	0.5%	0.02	0.07	0.08	0.04
C	0.3%	0.04	0.13	0.21	0.20
	0.5%	0.01	0.07	0.07	0.05
D	0.3%	0.08	0.10	0.27	0.31
	0.5%	0.12	0.13	0.25	0.22

Table 3.1: Errors of estimated t statistic thresholds.

Exhaustive Evaluations on ADNI

The results from simulations and the four real-world datasets have clearly shown that RapidPT estimates the max Null distribution with high fidelity and by only taking few orders of magnitude less time than NaivePT. In this section, we compare the algorithm to the more state-of-the-art version of permutation testing via SnPM (see Section 3.6). Since the crux of RapidPT comes from its sub-sampling and training/recovery modules, each governed by hyperparameters, in these third phase of the results we restrict to ADNI data and explore the landscape of all hyperparameters. The results still include accuracy and runtime gains, but with respect to SnPM we focus on resampling risk images for clinical significance.

Recovery of the Null vs. SnPM

Recovery is still good. We discuss one set of results here. The left plot of Figure 3.8 uses a colormap to summarize the KL divergence results obtained from comparing the max null distributions of a single run of SnPM versus multiple RapidPT runs with various hyperparameters. The right plot of Figure 3.8 puts the numbers displayed in the colormaps into context by showing the actual max null distributions for a single combination of hyperparameters. As we

have been observing until now, the sub-sampling rate was the hyperparameter that had the most (significant) impact on the KL divergence. A sub-sampling rate of 0.1% led to high KL divergence, i.e., the max null distribution was not recovered in this case. For every other combination of hyperparameters (as low as 0.35%), RapidPT was able to recover an accurate max null distribution. Most KL divergence values were in the $0.01 - 0.05$ range with some occasional values between $0.05 - 0.15$. The number of permutations used has an influence on the range of these KL values (we will see the influence of L later).

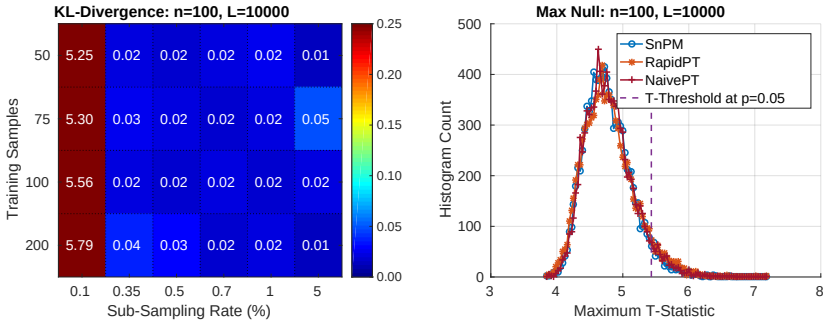


Figure 3.8: Recovery of RapidPT versus SnPM.

Are we picking up correct voxels?

The test statistics obtained using the original data labels (i.e., grouping labels) whose value exceeds the t-threshold associated to a given p-value will correspond to the null hypothesis rejected. Figure 3.9 shows the resultant mapping between t-threshold and p-values for the max null for a given set of hyperparameters. It is evident that the difference across methods is minimal. Moreover, Figure 3.9 also shows that the low p-value regime ($p < 0.1$), which is in some sense of more interest for practical purposes, is similar across all the methods. However, despite the low percent differences between the p-values, for larger datasets, with 100, 200, and 400 subjects here, RapidPT consistently yields slightly more conservative p-values near the tails of the distribution. This conservativeness implies that RapidPT will reject the null hypothesis for a slightly lower number of voxels than SnPM or NaivePT. A better way to figure out if this difference is significant is to simply look at the activation regions (voxels) that survive the test. Figure 3.11 shows this resampling risk between RapidPT and the two baselines. And some example regions picked up by both

RapidPT and SnPM are shown in Figure 3.10. Note that although the p-values agree across methods, even the slightest of differences may lead to slightly more or fewer null hypotheses to be rejected. This has a direct impact on the risk between RapidPT and classical permutation testing (NaivePT or SnPM).

Recall the definition of the risk from (3.9). In datasets where there is a small signal difference between the groups of interest to begin with, we may see an elevated resampling risk. This is because a very small number of null hypotheses will be rejected. Hence, at a given p-value, one of the methods will reject a small number of null hypotheses while the other will not reject the same set of hypotheses until the p-value is increased by some $\delta \ll 1$, thereby influencing resampling risk. For instance in Figure 3.11 with $n = 200$ at $p = 0.05$, RapidPT rejected 59 out of ~ 568k null hypotheses while SnPM rejected 71, resulting in a resampling risk of 8.45%. On the other hand, for $n = 400$ at $p = 0.05$ RapidPT rejected 2158 out of ~ 570k statistics and SnPM rejected 2241 null hypotheses, resulting in a lower resampling risk of 1.85% even though there is a larger difference in the number of rejected hypotheses. Nonetheless, as we can see in the brain maps in Figures 3.10 this difference of RapidPT and SnPM rejections were among the boundary voxels of the rejection regions i.e., the mismatch is not at the center of the significant region. Here the regions picked up correspond to the *corpus callosum* – which is one of the primary brain region that corresponds to the signature of cognitive decay at the onset of Alzheimer's disease. Note that once a reasonable small smoothing filter is applied to nullify the noise coming from stand-alone singleton significant voxels, this apparent risk will vanish. In summary, these results suggest that with respect to detectable activation regions in the brain, the approaches yield similar results – an aspect which if vital in practice.

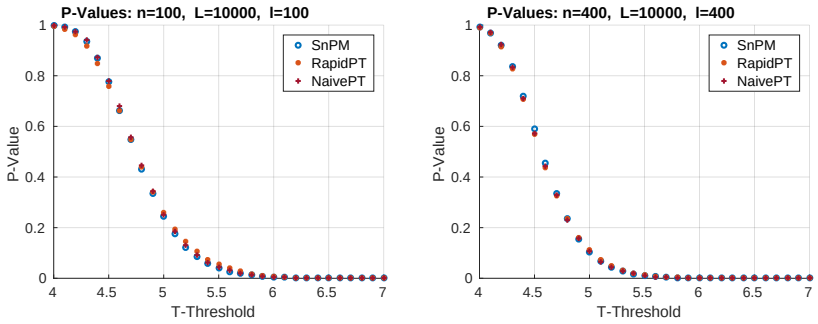
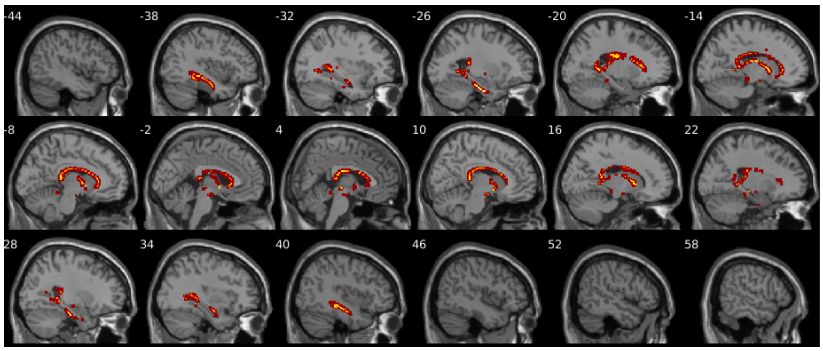
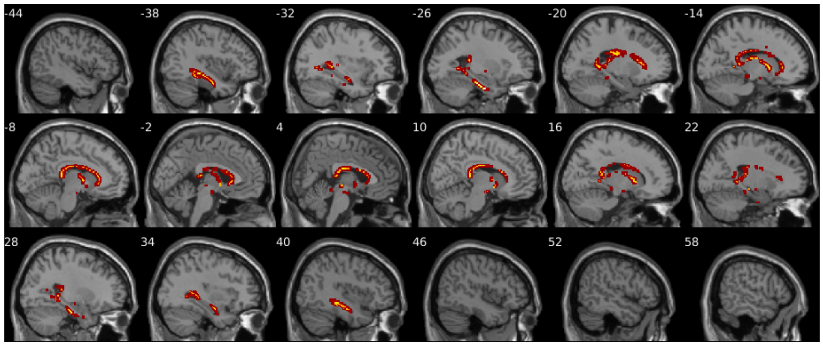


Figure 3.9: p-values for SnPM, RapidPT, and NaivePT.



(a) SnPM Statistic Maps. Slice -44 to 58 from left to right top to bottom.



(b) RapidPT Statistic Maps. Slice -44 to 58 from left to right top to bottom. $\eta = 0.5\%$, $\ell = 400$.

cluster-level	voxel-level				x,y,z mm		
k	$p_{FWE\text{-}corr}$	$p_{FDR\text{-}corr}$	T	p_{uncorr}			
82	**0.0000**	**0.0005**	**10.03**	0.0000	**-26**	**-12**	**-32**
	0.0000	0.0005	9.58	0.0000	-24	-24	-21
	0.0015	0.0005	5.84	0.0000	-30	-18	-26
68	**0.0000**	**0.0005**	**9.55**	0.0000	**27**	**-22**	**-21**
	0.0000	0.0005	8.96	0.0000	26	-10	-32
1372	**0.0000**	**0.0005**	**9.04**	0.0000	**-33**	**-9**	**-20**
	0.0000	0.0005	8.38	0.0000	-38	-28	-9
	0.0000	0.0005	7.97	0.0000	12	-6	27
173	**0.0000**	**0.0005**	**8.74**	0.0000	**36**	**-6**	**-22**
	0.0000	0.0005	7.85	0.0000	38	-34	-4
	0.0000	0.0005	7.51	0.0000	39	-15	-16
21	**0.0000**	**0.0005**	**7.67**	0.0000	**-21**	**-36**	**3**
75	**0.0000**	**0.0005**	**7.59**	0.0000	**14**	**9**	**0**
	0.0000	0.0005	6.55	0.0000	9	4	-6
	0.0001	0.0005	6.25	0.0000	18	12	6
15	**0.0000**	**0.0005**	**7.51**	0.0000	**0**	**0**	**2**
23	**0.0000**	**0.0005**	**6.76**	0.0000	**28**	**-40**	**-6**
	0.0000	0.0005	6.55	0.0000	26	-46	0

Height threshold: statistic u= 5.16 (0.0500 FWE) Degrees of freedom = [1 398]
Design: 2 Groups: Two Sample T test; 1 scan per subject: 200(GrpA),200(GrpB)
Search vol: 1939410 cmm, 574640 voxels
Perms: 100000 permutations of conditions, bhPerms=0 Voxel size: [1.50, 1.50, 1.50] mm

(c) SnPM Report

cluster-level	voxel-level				x,y,z mm		
k	$p_{FWE\text{-}corr}$	$p_{FDR\text{-}corr}$	T	p_{uncorr}			
79	**0.0000**	**0.0004**	**10.03**	0.0000	**-26**	**-12**	**-32**
	0.0000	0.0004	9.58	0.0000	-24	-24	-21
	0.0042	0.0004	5.84	0.0000	-30	-18	-26
65	**0.0000**	**0.0004**	**9.55**	0.0000	**27**	**-22**	**-21**
	0.0000	0.0004	8.96	0.0000	26	-10	-32
1218	**0.0000**	**0.0004**	**9.04**	0.0000	**-33**	**-9**	**-20**
	0.0000	0.0004	8.38	0.0000	-38	-28	-9
	0.0000	0.0004	7.97	0.0000	12	-6	27
164	**0.0000**	**0.0004**	**8.74**	0.0000	**36**	**-6**	**-22**
	0.0000	0.0004	7.85	0.0000	38	-34	-4
	0.0000	0.0004	7.51	0.0000	39	-15	-16
20	**0.0000**	**0.0004**	**7.67**	0.0000	**-21**	**-36**	**3**
63	**0.0000**	**0.0004**	**7.59**	0.0000	**14**	**9**	**0**
	0.0001	0.0004	6.55	0.0000	9	4	-6
	0.0006	0.0004	6.25	0.0000	18	12	6
15	**0.0000**	**0.0004**	**7.51**	0.0000	**0**	**0**	**2**
9	**0.0000**	**0.0004**	**7.45**	0.0000	**-27**	**-40**	**-8**
21	**0.0001**	**0.0004**	**6.76**	0.0000	**28**	**-40**	**-6**
	0.0001	0.0004	6.55	0.0000	26	-46	0

Height threshold: statistic u= 5.25 (0.0500 FWE) Degrees of freedom = [1 398]
Design: 2 Groups: Two Sample T test; 1 scan per subject: 200(GrpA),200(GrpB)
Search vol: 1939410 cmm, 574640 voxels
Perms: 100000 permutations of conditions, bhPerms=0 Voxel size: [1.50, 1.50, 1.50] mm

(d) RapidPT Report

Figure 3.10: ADNI Statistic Maps, SnPM vs. RapidPT

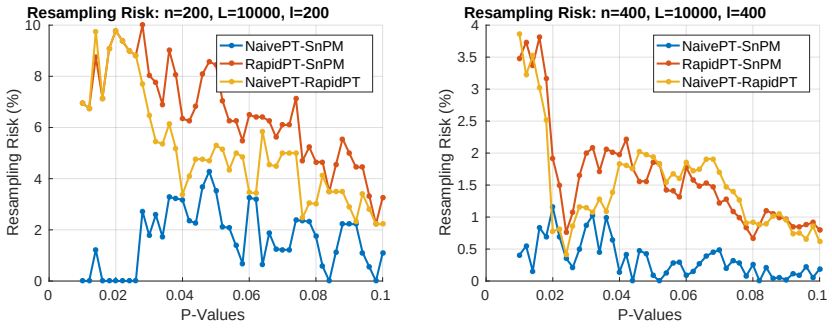


Figure 3.11: Resampling risk of NaivePT, SnPM and RapidPT.

Effect of hyperparameters on runtime performance of RapidPT

The main advantage of RapidPT is its runtime gains compared to alternative approaches, as was shown earlier in Section 3.7 and 3.7. Building upon this, Figures 3.12 shows the speedup gains of RapidPT over SnPM for some of the exhaustive evaluations on ADNI data. Column corresponds to a single dataset size and rows correspond to different number of permutations L. RapidPT outperforms SnPM in most scenarios, With very few exceptions, the colormaps show that RapidPT is $1.5 - 30$x faster than SnPM. As expected, a low η (0.35%,0.5%) and l ($\frac{n}{2}$,$\frac{3n}{4}$) leads to the best runtime performance without a noticeable accuracy tradeoff (see Figure 3.8 from earlier in this section).

Scaling up L **and** n

As opposed to η and ℓ which only seem to have an impact on the runtime performance of RapidPT, the number of permutations L and the size of the dataset n have an impact, unsurprisingly, on the runtime of both RapidPT and SnPM. Hence, we conclude the exhaustive set of results in this section by evaluating the scalability of both RapidPT and SnPM versus L and n. Figures 3.13 and 3.14 show these results ($\eta = 0.35\%$ and $\ell = n$ here). Figure 3.13 shows the super-linear scaling of SnPM compared to RapidPT as L increases. Doubling L in SnPM leads to an increase in the runtime by a factor of about two, on the other hand, doubling L for RapidPT only affects the runtime of the recovery phase leading to smaller change in the runtime performance. The runtime of both models is only comparable if we focus on the lower range of the number of permutations (5000) across different datasets (n). As this L increases (the more *practical* setting) and as shown clearly in Figure 3.12, RapidPT outperforms the

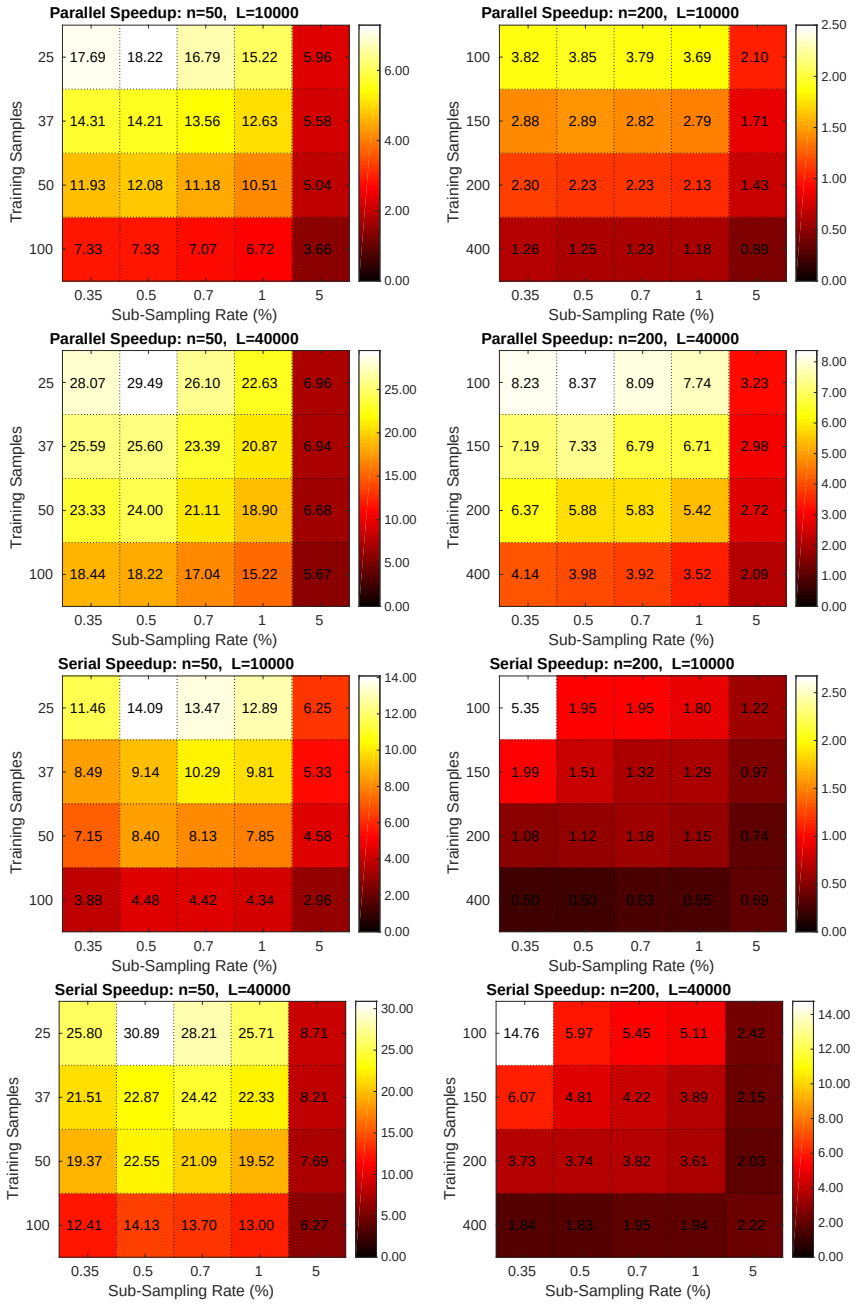


Figure 3.12: Colormaps of the speedup gains of RapidPT over SnPM.

permutation testing implementation within SnPM. Figure 3.14 shows the effect of the dataset size. For SnPM, as expected, the runtime scales up approximately linearly with increasing n. For RapidPT, however, increasing n has a variable effect on the runtime. The training phase ends up contributing more to the runtime since we chose ℓ to be proportional to n (see Section 3.6), while the recovery phase runtime increases at much slower rate. In summary, scaling the number of permutations has a stronger impact on the runtime performance of SnPM than RapidPT. On the other hand, scaling the dataset size has a more detrimental effect on the timing of RapidPT than for SnPM. Furthermore, if both parameters are increased at the same time the runtime of SnPM increases at a much faster rate than the runtime of RapidPT – making the proposed faster permutation testing model the desirable candidate.

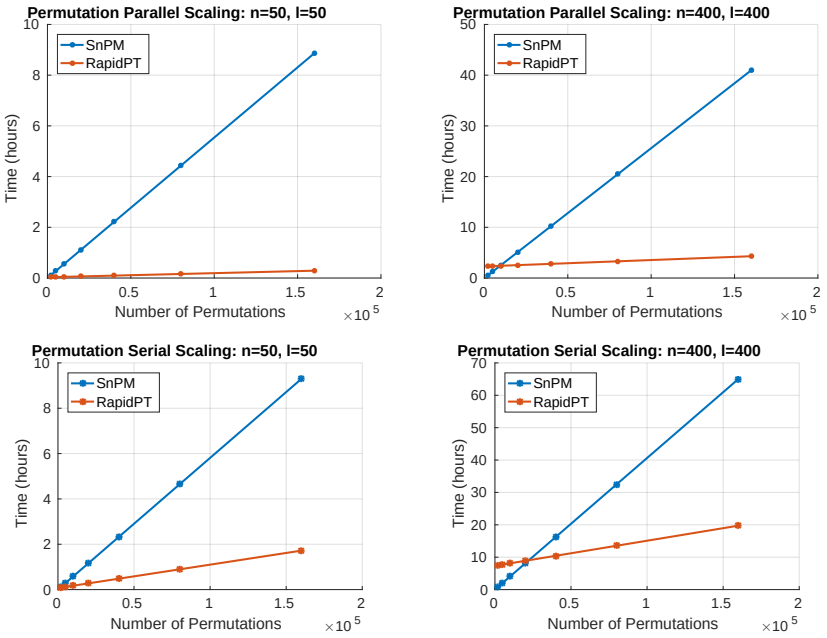


Figure 3.13: Effect of L on runtime performance of RapidPT and SnPM.

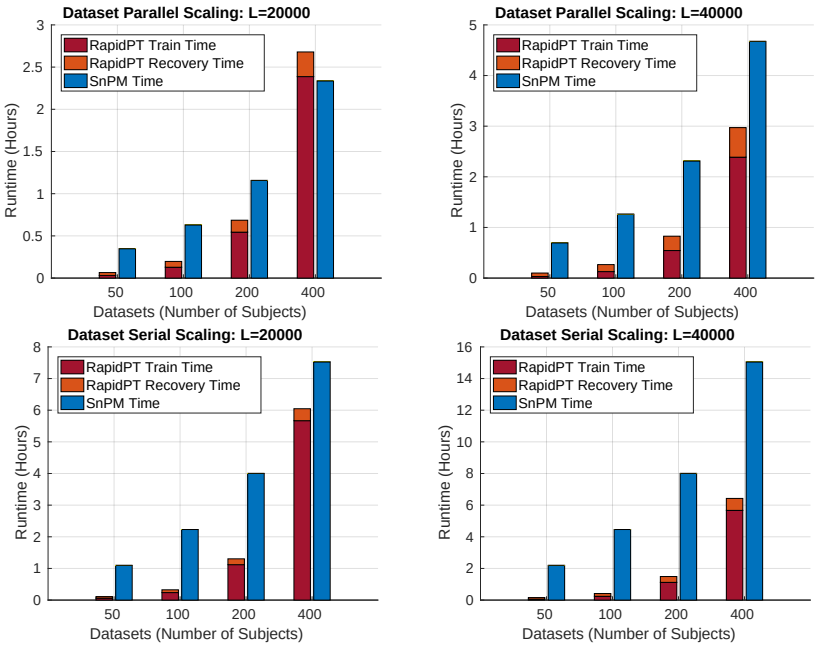


Figure 3.14: Effect of the dataset size on RapidPT vs. SnPM.

3.8 Discussion

Hyperparameter Recommendations

Sub-sampling Rate. Our experiments showed that as long as we selected a sub-sampling rate, $\eta \geqslant 0.35\%$, we could accurately recover the max null distribution. This value directly follows from the theory of LRMC as well as the analysis of recovery guarantees from Section 3.5. In particular, the authors of (Candès and Tao, 2010) and others (the low-rank recovery theory in general) have shown that given a *sufficient* number of entries one can recover the orthogonal basis of the row space as well as the expansion coefficients. This Nyquist number of entries is on the order of $r\log(d)$, where r is the column space rank and d is the ambient dimension. In our case, the column rank space of $\mathbf{T}$ is bounded by the number of subjects (i.e $r \leqslant n$) by definition, and in general the ambient dimension of $\mathbf{T}$ is v. An estimate of the minimum sub-sampling rate is then given as follows.

$$\eta \geqslant \frac{n\log(v)}{v} \tag{3.10}$$

Recall the KL-Divergence results from Figures 3.4 and 3.8, where we see that as long as η is greater than or equal to a certain threshold, RapidPT is able to accurately recover the max null distribution. The simulation results in Figures 3.4 and 3.6 also explicitly show a certain such threshold. For the simulation study this was $\approx 1.6\%$, and evidently (3.10) is little conservative because as shown in our results we were able to recover the maxnull for all of the datasets with $\eta \geqslant 0.35\%$. In practice we set it to a default conservative value of $2 * \eta_{\mathtt{min}}$, which is also shown in Figures 3.4 and 3.6. This slightly larger choice ensures the error in recovering the null is almost negligible. A user does not need to change this rate in practice beyond what is given by the toolbox (see Section 3.8). However, if they are willing to be more flexible by sacrificing some accuracy to achieve an even higher speed-up, η can be reduced appropriately. Overall, the accuracy of RapidPT will not significantly improve as the sub-sampling rate increases beyond this $\eta_{\mathtt{min}}$. On the other hand, a low sub-sampling rate can significantly reduce the runtime, in particular in the large $\mathtt{L}$ regime, and is therefore preferable.

Number of Training Samples. The number of training samples will ideally be the exact rank of $\mathbf{T}$, which is not known, however, recall that the rank is bounded by the number of subjects in the data matrix used to generate $\mathbf{T}$. In our evaluations, we are able to accurately recover the max null distribution even when using as low as $\frac{\mathtt{n}}{2}$ training samples. Nevertheless, the recommended number of training samples in practice is $\mathtt{n}$, and is the default setting within RapidPT toolbox (see Section 3.8).

Number of Permutations $\&$ **Dataset Size.** When a large number of permutations is desired, RapidPT should be strongly considered due to its runtime gains. Beyond the order of magnitude or more speed-up (compared to using SnPM), the accuracy will improve as well as the KL-Divergence results show. Although not a hyperparameter, the dataset size $\mathtt{n}$ should play a key role when the user selects which method to use – RapidPT or SnPM. For large datasets ($\mathtt{n} \geqslant 200$ or so), RapidPT will be a good option if the user is planning to perform a large number of permutations. For medium-sized datasets ($50 \leqslant \mathtt{n} \leqslant 200$ or so), RapidPT will most likely lead to good speedup gains. For small datasets ($\mathtt{n} \leqslant 50$) on the other hand, although RapidPT might lead to speedup gains over regular permutation testing, the total runtime will be on the order of minutes anyway – so it really does not matter what method we choose. Further, as

we briefly discussed in Section 3.1, (Winkler et al., 2016) does provide several strategies for reducing the runtime of permutation testing keeping sizes of datasets in mind as well. But the authors do not report significant speedup gains against regular permutation testing on a 50 subject dataset with ≈ 200k voxels. On the other hand, RapidPT is able to consistently outperform the state of the art permutation testing implementation (SnPM) on a 50 subject dataset with ≈ 540k voxels. This boost in performance is, mainly, due to our low sub-sampling rates followed by robust subspace tracking recovery.

SnPM Integration

SnPM is a toolbox that can be used within the software Statistical Parametric Mapping (SPM) (SPM, 2012). RapidPT has been integrated into the next development version of SnPM (SnPM, 2013). This enables users to leverage the pre-processing and post-processing capabilities of SnPM. Through the graphical user interface (GUI) of SnPM the user can simply specify if they want to use RapidPT. Alternatively, an experienced user can also toggle a flag called *RapidPT* inside the *snpm_defaults.m*. Once this flag is set the user can simply proceed with their normal SnPM workflow. The SnPM GUI does not allow the user to set RapidPT's hyperparameters (η, l), however, the online documentation walks the user through the process of setting them manually. Further discussion about the usage of both SnPM, and RapidPT module within SnPM can be found in the documentation of the respective toolboxes. The RapidPT library webpage is at http://felipegb94.github.io/RapidPT/.

Chapter 4

Robust Sample Enrichment for Clinical Trials

While the choice of a statistical summary that measures the difference between placebo and treatment groups is one vital component of trial design, the other critical aspect is to figure out whether to include an individual in the trial. Enriching the trial population to include appropriate subjects is the focus of this chapter. We will begin with motivating the need for trial enrichment, specifically from the perspective of Alzheimer's disease research.

4.1 Introduction

As we discussed earlier in Section 2.4, various groups have adapted sophisticated machine learning methods, to *learn* patterns of pathology by classifying healthy controls from AD subjects. The success of these methods ($> 90\%$ accuracy (Zhang et al., 2011b)) has led to attempts at more fine grained classification tasks, such as separating MCIs, and even identifying which MCI subjects will go on to develop AD (Teipel et al., 2007; Hinrichs et al., 2011a). Even in this difficult setting, multiple current methods have reported over 75% accuracy. While accurate classifiers are certainly desirable, one may ask if they address a real practical need — *if no treatments for AD are currently available, is AD diagnosis meaningful?* To this end, (Hinrichs et al., 2012a; Kohannim et al., 2010) showed the utility of computational methods beyond diagnosis/prognosis; they may in fact be leveraged for designing *efficient* clinical trials for AD. There are however several issues with such procedures – even after choosing a robust outcome measure that quantifies the trial groups, and this work is in this context of designing methods for AD trials which are *deployable in practice* and *cost-effective*. We begin by looking at the current landscape of MCI-based trials.

Recent trials designed to evaluate new treatments and interventions for AD at the mild to moderate dementia stage have largely been unsuccessful and there is growing consensus that trials should focus on the earlier stages of AD including MCI or even the presymptomatic stage (Grill et al., 2013; Grill and Monsell, 2014), if such stages can be accurately identified in individual subjects (Jelic et al., 2006; Petersen, 2007; Aisen, 2011). More than 500 clinical trials have been conducted since early 2000s, and as of Dec 2017, ClinicalTrials.gov lists 230 active (recruiting) clinical trials with 27 in Phase 3 and 4. At least 68 of these trials are in Phase 3 or more (see Section 2.1). However, 80% or more of these trials have failed mainly because of the lack of detectable signal difference between placebo and treatment. Apart from the fact that the drug may not be inducing pathological/therapeutic changes in the subject's health, there is a critical issue from the perspective of trial design itself. This is because, MCI is a clinical syndrome with heterogeneous underlying etymology that may not be readily apparent from a simple diagnostic work-up, posing a major challenge in reliably identifying the most probable beneficiaries of a putative effective treatment (Albert et al., 2011a). For instance, MCIs may have clinical but not biomarker evidence of incipient AD, may have biomarker evidence in some modalities, or, despite biomarker presence, may not show symptomatic progression during the trial time-period. An efficient MCI trial would ideally include only those patients most likely to benefit from treatment; who possess AD pathology based on a constellation of amyloid, tau and neural injury biomarker assessments, and who are most likely to progress clinically to symptomatic AD.

However, the typical annual conversion rate to dementia among MCI due to AD is $3-20\%$ across several studies (Mitchell and Shiri-Feshki, 2009). Hence, over a two year trial, at best only 40% of participants would have naturally progressed, and the ability to detect the true efficacy of the trial is perhaps diminished – even before accounting for the ability of a drug to improve the health. Several ongoing AD trials *enrich* their population using one or more disease markers as inclusion criteria (Grill and Monsell, 2014; Lorenzi et al., 2010). The general framework here is to effectively screen out subjects who are weak decliners (i.e., MCI who may not convert to AD) (Leoutsakos et al., 2014). Unless there is a natural phase change (i.e., an elbow) in the distribution for distinguishing the at-risk and not-at-risk subjects, a fixed fraction of the total cohort are filtered out based on the study design. Imaging-based markers (e.g., Fluorodeoxyglucose (FDG), hippocampal and ventricular volume) and cerebrospinal fluid (CSF) profiles have been shown to be effective in screening out

low-risk subjects, due to the fact that disease manifests much earlier in imaging data compared to cognition (Grill et al., 2013; Grill and Monsell, 2014). However, these markers are uni-modal while several studies have shown the efficacy of multi-modal data (Hinrichs et al., 2011a; Zhang et al., 2011b). Furthermore, CSF cannot be used in practice as a screening instrument because assays typically need to be performed in a single batch and are highly lab specific (Mattsson et al., 2013).

An alternate approach is to use *computational* multi-modal markers derived from support vector machines (SVMs) and other machine learning models (Lorenzi et al., 2010; Hinrichs et al., 2012a; Kohannim et al., 2010; Escudero et al., 2011; Yu et al., 2014). The strategy here uses imaging data from two time points (i.e., TBM or hippocampus volume change) and derives a machine learning based biomarker. Based on this marker, say, the top (strongest decliners) one–third quantile subjects may be selected to be included in the trial. Using this enriched cohort, the drug effect can then be detected with higher statistical power, making the trial more cost effective and far easier to setup/conduct. Most such approaches use longitudinal data, however, a practical enrichment criterion should only use baseline (trial start-point) data. We argue that existing approaches to enrichment, including state-of-the-art computational techniques, *cannot* guarantee this optimal enrichment behavior – optimally correlate with dementia spectrum with high confidence having access only to the baseline data, while simultaneously ensuring small intra-stage variance. Instead, we approach the optimal enrichment design from basic principles. Consider a RCT where participants are randomly assigned to treatment and placebo groups (see Section 2.1), and the goal is to quantify any drug effect. Recall that if the distributions of the trial outcome for the two groups are statistically different, we conclude that the drug is effective. Clearly when the effects are subtle, the number of subjects required to see statistically meaningful differences can be huge, making the trial infeasible. Note here that, using fast hypothesis testing procedure proposed in Chapter 3, customized outcomes may also be derived that capture subtle (weak) signals by assigning predictions based on probabilities of class membership.

If the classical outcomes, or the more customized predictions, are statistically well-separated in some sense, it directly implies that potential improvements in power and the efficiency of the trial are possible. This first, fundamental aspect, we show in this work is that that high statistical power in these experiments is not merely a function of the classification accuracy of the model, rather the

conditional entropy of the outputs (prediction variables) from the classifier at test time. In other words, an increase in classifier accuracy does not directly reduce the variance in the predictor. This insight is based on a simple observation that the efficacy of a statistical test is directly related to the coefficient of variation of the statistic that measures the confidence of hypothesis (as we will discuss in later sections). Therefore, SVM type methods *are* applicable, but significant improvements are possible by deriving a learning model with the *concurrent* goals of classifying the stages of dementia *as well as* ensuring small conditional entropy of the outcomes. We then focus on designing specialized learning architectures towards this final objective – the aspect of *taking into account the output variance of trial outcome while choosing the study population*.

We achieve the above goals by proposing a novel machine learning model based on ideas from deep learning (Bengio et al., 2015). Deep architectures are non-parametric learning models (Bengio, 2009b; Bengio et al., 2013b; Erhan et al., 2010a) that have received much interest in machine learning, computer vision and natural language processing recently (Kavukcuoglu et al., 2010; Krizhevsky et al., 2012; Dahl et al., 2011). They are effective and robust in learning complex concepts, and recently several authors extensively studied their success from both empirical and theoretical view-points (Bach, 2014; Ithapu et al., 2015a). Although powerful, it is well known that they require very large amounts of data (unsupervised or labelled) (Bengio, 2009b), which is infeasible in medical applications including neuroimaging, bioinformatics etc., where data dimensionality (d) is always much larger than the number of instances (n). A naïve use of off-the-shelf deep networks expectedly yields poor performance. Nevertheless, independent of our work, deep learning was used in structural and functional neuroimaging, where (Suk and Shen, 2013) use a region of interest approach while (Gupta et al., 2013; Plis et al., 2013) sub-samples each data instance to increase n. This work provides a mechanism where no such adjustments are necessary, and, in fact, the framework developed here is more generally applicable for learning deep networks in the small sample regime, i.e., learning problems where number of data instances is much smaller than the data dimensionality like whole–brain voxel–wise analysis.

The **contributions** of this chapter are as follows: **(a)** We propose a scalable, and a general, deep learning framework that is applicable for learning problems in the small sample regime, and provide certain guarantees for their performance; **(b)** Using our proposed models, we design novel predictive multi-modal imaging-based disease markers, based only on the trial start-time (baseline)

acquisitions, that correlates very strongly with future AD decline; and **(c)** We show via extensive analyses using imaging, cognitive and other clinical data that the new computational markers result in cost-efficient clinical trials with moderate sample sizes when used as trial inclusion criteria. Section 4.2 briefly introduces clinical trials and deep networks, and Section 4.3 presents the optimal enrichment design. Sections 4.4 presents our proposed models, referred to as *randomized deep networks*. Sections 4.5 and 4.6 extensively evaluate these models and discuss their efficacy in enrichment. The contributions of this chapter are presented in Ithapu et al. (2014b,a) and Ithapu et al. (2015b).

4.2 Preliminaries

We first present some preliminaries on the sample size estimation for conducting clinical trials (Friedman et al., 2010) discussing the necessity of a computational enrichment criterion. We also discuss briefly (to the extent that is needed in this chapter) the design and learning of neural networks, which is critical for the proposed enrichment criteria. Although there are many classes/types of deep architectures in the literature, we focus our presentation on fully connected designs using stacked denoising autoencoders (Vincent et al., 2010) and fully-supervised dropout (Srivastava et al., 2014). This is primarily because of the nature of the dataset/task that we are concerned with here (as discussed in Section 4.2 below). Nevertheless, the ideas presented here are applicable for any architectures used for learning problems in the small-sample regime.

Sample enrichment

Consider a randomized controlled trial (RCT) designed to test the efficacy of some treatment for an underlying disease condition (Friedman et al., 2010; Sakpal, 2010). Recall the setup of RCT from Section 2.1. The population under study are randomly assigned to either of the treatment (or intervention) and non-treatment (or placebo) groups. If the drug indeed offers an improvement, then the two groups should show this change when measured using some outcome measure summarizing the disease status. Such change would generally correspond to *reducing* the disease progression to a certain extent, referred to as the effect size (Sakpal, 2010), in the treatment group compared to the placebo group. Effect size $\eta \in [0, 1]$ is simply a way of quantifying the size of the

difference between two groups and is easy to interpret [1]. For example, an effect size of 0.25 for the drug implies that the treatment group is 25% *better* than the placebo group – the incipient disease in treatment group is 25% lower than that in the placebo group. As we discussed in Section 2.1, the disease is quantified via the trial outcome which, in general, is a reliable disease marker. One can use the ideas presented in Chapter 3 to design such an outcome, or it may come from alternate studies/sources.

Given such an outcome, the trial efficacy is measured by estimating the Type-II error between the two groups, after inducing the drug or intervention. Clearly, this Type-II error (and the effect size) would be influenced by the choice of the outcome and the size of the trial population. Hence, in practice, one would want to 'estimate' the trial's efficacy ahead of time to ensure that the effect size is good enough, the population is reasonably large and diverse, and the outcome is appropriately chosen – basically ensuring that the trial makes sense. Measuring this efficiency of a trial then requires simulating a hypothetical RCT. Here, the drug is induced by *fixing* the effect size ahead of time, and computing the resulting Type-II error for the given outcome and population. Let δ denote the difference of mean outcome (the standard change) between the trial start and end points (e.g., 2 years) in the placebos. Let σ be the standard deviation of the outcome, and the effect size be η (e.g., 0.25, a 25% reduction in the disease is desired). δ and σ are known apriori (reported in alternate studies on the disease). The treatment group is then *expected* to have the change in outcome decreased to $(1-\eta)\delta$, which will correspond to a hypothetical improvement of η induced by the drug/treatment. Within this setting, the number of samples s required per arm (treatment and placebo) is given by (Friedman et al., 2010)

$$s = \frac{2(Z_\alpha - Z_{1-\beta})^2\sigma^2}{(1-\eta)^2\delta^2} \tag{4.1}$$

where $(1-\beta)$ denotes the desired statistical power at a significance level of α. This expression directly follows from applying a difference of means t-test, where the means are computed from the two distributions of interest – outcome change in treatment and outcome change in placebo respectively (Friedman et al., 2010). The Null hypothesis for this t-test is that mean change in the outcome is same for the treatment and placebo groups. The necessity of sample enrichment can be directly seen from (4.1). If the population under study has less standard change in the outcome δ, then the required sample s

[1] `https://www.leeds.ac.uk/educol/documents/00002182.htm`

for achieving a given power $1-\beta$ will be very large. Alternatively, the power can be maximized only when the incipient change δ is as large as possible in the trial population. Hence, the participants need to be *enriched* so as to include only those subjects with large change in the disease *during* the trial time-line. We define an *optimal enrichment criterion*, in chapter-4-section 4.3, based on these ideas and (4.1). It should be noted that there are alternate approaches to computing the sample sizes in clinical trials, depending on the RCT structure, and the choice of the statistic test to measure the group difference (Friedman et al., 2010). However, the dependence of η and δ on the sample size is similar across all such sample size calculations.

Neural Networks

Artificial neural networks (ANN) are representation learning machines introduced in the late 50s for learning complex concepts (Minsky and Papert, 1969). An ANN transforms a given (input) instance/example (a d-dimensional vector of features/covariates) into a new representation that may be used within the context of classification or regression or other learning paradigms. These transformations are non-linear, and possibly non-convex, and calculated by first computing an affine projection of the input, followed by applying a monotonic non-linear 'activation' function over these projections (which may not be necessarily point-wise) (Bengio, 2009b). The resulting features correspond to some high-level and abstract representations of the inputs (Bengio et al., 2013b, 2015; Wang and Raj, 2015). The necessity of using non-linear and possibly discontinuous activation functions has been well documented, both empirically (Bengio et al., 2015; Nair and Hinton, 2010) as well as theoretically (Bianchini and Scarselli, 2014; Arora et al., 2014). Given a set of examples $(\mathbf{x}_i, \mathbf{y}_i)_{i=1}^n$ from n different instances (or subjects in the case of medical research), the new representations are given by $\mathbf{h}_i = \sigma(\mathbf{W}\mathbf{x}_i + b)$. $\mathbf{W} \in \mathbb{R}^{d_1 \times d}$ and $b \in \mathbb{R}^{d_1 \times 1}$ are the unknown transformation coefficients (i.e., weights) (assuming $y_i \in \mathbb{R}^{d_1}$). $\sigma(\cdot)$ is the point-wise activation. Examples of $\sigma(\cdot)$ include sigmoid/logistic function (i.e., $\sigma(z) = \frac{1}{1+\exp(-z)}$), hyperbolics, rectified linear units (Dahl et al., 2013) etc. Figure 4.1 plots the list of widely used activation functions. Observe that the choice of the non-linearity is almost entirely empirical (using domain knowledge).

This *single-layer* neural network comprises of a visible layer ($\mathbf{x}_i$s) and an output layer (y_is). *Multi-layer* neural networks (MLNN) perform $L > 1$ such

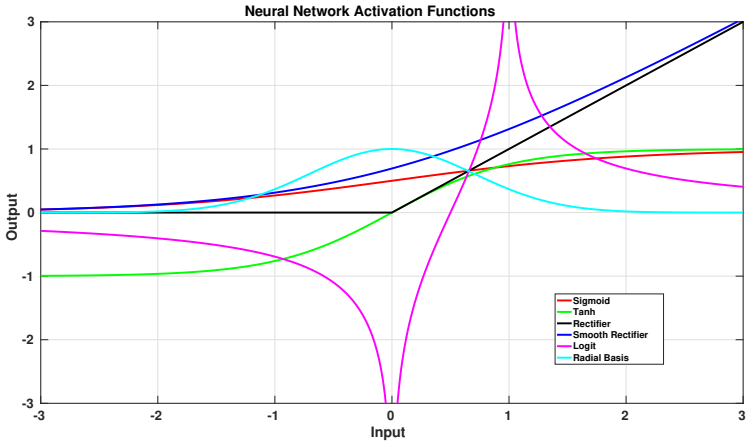


Figure 4.1: Activation functions used in neural networks

transformations sequentially, thereby resulting in L *hidden* layers ($\mathbf{h}_i^\ell$s, $\ell = 1, \ldots, L$).

$$\mathbf{h}_i^\ell = \sigma(\mathbf{W}^\ell \mathbf{h}_i^{\ell-1} + b^l) \quad \mathbf{h}_i^0 = \mathbf{x}_i \quad \ell = 1, \ldots, L \quad i = 1, \ldots, n \tag{4.2}$$

The highest level hidden layer ($\mathbf{h}_i^L$s) may then be used for a given learning task. Specifically, a classification or regression model can be trained using $(\mathbf{h}_i^L, y_i)_{i=1}^n$, or in certain situations, $\mathbf{h}_i^L$s may directly correspond to y_is, Once the learning is done, the output layer of MLNN would simply be the predictions $\hat{y}_i$s. The underlying hypothesis of multi-layer networks is that, multiple, sequential, non-linear compositions of the input data allows for manipulating the highest-order interactions in the input features to construct a sequence of hidden representations $\mathbf{h}_\cdot^\ell$ ($\ell = 1, \ldots, L$). Several classical, and recent, works have provided extensive empirical justification, as well as theoretical motivation for constructing such multi-layer networks (Bengio et al., 2015; Wang and Raj, 2015; Szegedy et al., 2014). The hidden layer lengths are denoted by $d_\ell, l = 1, \ldots, L$, and whenever $d_\ell > d$ for some ℓ, the network projects the data into high-dimensional, learns over complete representations of the inputs, and then projects them into the appropriate output space of dimensions. Figure 4.2(a) shows the architecture of a typical L-layered MLNN. Layer to layer connections shown in the figure correspond to applying the corresponding

transformations $((\mathbf{W}^l, b^l))$, followed by point-wise non-linearity $\sigma(\cdot)$.

Backpropagation and Deep Learning

To learn the unknown transformations $(\mathbf{W}^l, b^l)$ one compares the MLNN predictions $\mathbf{y}_i$s to the desired outputs using some appropriate loss function $\mathcal{L}(\cdot)$ chosen entirely based on the problem at hand (e.g., squared loss for regression, logistic/entropy loss for classification, or some form of divergence). This objective is clearly non-convex, because of the presence of multiple (nested, L number of) composition of non-linear, possibly non-convex functions. Hence, there are no-guarantees in optimizing the loss function to approach to the global minimum. Instead, all we can rely on is stochastic gradient descent (SGD) (Bottou, 2010), and the instantiation of SGD for neural networks is referred to as *backpropagation* (LeCun et al., 2012). The name comes from the fact that the errors in the final/output layer need to be 'propagated' back to the inputs for estimating the coefficients. Several variants of back-propagation have been proposed over the last few decades (see (LeCun et al., 2012; Bengio et al., 2015)). It is impractical to review the vast literature on various back-propagation strategies, and so, we only point out the main bottlenecks of the algorithm that eventually made neural network learning obsolete. An interested reader can refer to (Bengio et al., 2015; Bishop, 2006) and others for further details.

Although MLNNs have attractive theoretical properties, for instance a 3-layer network can represent any polynomial of arbitrarily high degree given enough training data and computational time, see Chapter 5 in (Bishop, 2006), learning them involves a difficult optimization task due to the composition of the activations. It should be mentioned that even optimizing composition of linear maps may be non-trivial (Saxe et al., 2013). The parameter space is highly non-convex with many local minima and saddle points (Bottou, 2010) (see Chapter 6, (Bengio et al., 2015)). Stochastic minimization methods, nevertheless, compute a local minima, but these may be sensitive to initialization (Bengio, 2009b; LeCun et al., 2012). The *goodness* of such solutions – in terms of stability to noise, saddle point behavior and sensitivity to perturbations – depend on the number of stochastic iterations, and in turn, on the number of training samples available to exhaustively (and empirically) search through the solution space. Further, gradient based methods are prone to exponential decay of errors for large MLNNs, since the errors computed at the last layer 'die-out', in some sense, by the time they reach the inputs because of the presence of

non-linearities. Hence, even large errors from the objective may not lead to reasonably good gradient paths for the bottom layer transformations. Lastly, MLNN learning involves critical modeling/design choices about the network structure (number and lengths of layers).

Although several studies have shown the necessity of over-represented multi-layer networks, there was no consensus on which variant of classical back-propagation would best suit a given choice of network, and if there was any standardized way of choosing the network structure. These issues make efficient and/or robust learning of MLNNs difficult, which will eventually lead to poor generalization. Deep learning (a.k.a neural networks 2.0) refers to a suite of algorithms for efficient training of MLNNs by mitigating some of these issues (Bengio, 2009b; Bengio et al., 2015). The *depth* here simply refers to the many levels of transformations. The main issue with MLNNs is that multiple function compositions makes the search space highly non-convex with many local minima and saddle points. With large enough training instances and very high computational load (and using few innovative tricks to regularize the loss), the new wave of deep learning algorithms aim to steer through this complex landscape to estimate good (reproducible) solutions.

The earlier versions of deep learning algorithms partly mitigate the optimization issue by disentangling the compositions and only working with one-layer (one transformation) at a time. Working with each layer individually makes the objective simpler albeit, non-convex but likely with fewer local minima. Once *good* estimates of the transformations are obtained *layer-wise*, the entire network can then be initialized with these estimates. The resulting final layer predictions can then be compared with desired outputs to adjust or *fine-tune* the estimated parameters across all layers. This fine tuning is the same as performing complete back-propagation but using layer-wise estimates as an initialization or warm-start (Bengio, 2009b). This overall procedure is referred to as unsupervised pretraining, and unlabeled data are used to compute the layer-wise initialization. The challenge then is to construct such efficient layer-wise procedures, and several such schemes have been proposed, all of which fall under two families – restricted Boltzmann machines (Le Roux and Bengio, 2008) and autoencoders (Vincent et al., 2010). The basic rationale is that once reasonable layer-wise warm-starts are obtained, the transformation coefficients are already in some good solution bowl in the gradient search space, which may be smoother and better behaved than the ones obtained with random initializations (and/or learning all layers concurrently) or whole-network warm

starts (Bengio, 2009b).

Followed by unsupervised pretraining, a break-through to overcome over-fitting issues in backpropagation has been proposed. Over-fitting has been a critical aspect for learning large neural networks, which has not been addressed to any reasonable extent via pretraining. This algorithm is called *Dropout* (Srivastava et al., 2014), and the idea is to update only a fraction of the network's unknowns in each iteration of stochastic gradients. It is inspired from variance reduction and ensemble averaging (Dietterich, 2000), and we discuss this more in Section 4.2. More recently, (Ioffe and Szegedy, 2015) have discussed approaches to better regularize dropout training using batch normalization. This refers to re-normalizing instances within each batch and with each layer before using them to compute gradients. Although there is yet no theoretical justification, this has been shown to reduce covariance shift in backpropagation. Observe that unsupervised pretraining, dropout and batch normalization are approaches to improve backpropagation, and are independent of which type of architecture is being trained – fully connected or convolutional or recurrent (Jozefowicz et al., 2015). As discussed earlier, we will restrict our presentation to the non-convolutional architectures because of the nature of the data and task under investigation, which we now discuss below.

Fully Connected vs. Convolutional Designs: The topology (layer-to-layer connectivity) of a neural network plays a significant role in its generalization capacity. Broadly there are two such families of neural network architectures – fully connected and convolutional [2]. In fully connected layers, a linear transformation is first applied to the bottom layer units, followed by a point-wise non-linearity. In convolutional design, the bottom layer is convoluted with one (or more) filters followed by a non-linearity or a dimension reduction pooling (Nagi et al., 2011). The size of such filters is much smaller than the number of units themselves, thereby resulting in a higher layer that explicitly summarizes information from localized units from the bottom layer. Although both fully connected and convolutional designs have their merit, the convolutional setup is partly inspired from neuroscience studies about the nature/structure of animal visual cortex (Rosenblatt, 1961). Further, because of localization of signals, convolutional layers (in tandem with pooling) provide invariance to scaling and affine transformations. However, for the data sources we look at in this report –

[2] It is worth mentioning that a special case of convolutional design will correspond to fully connected connectivity, however, it is convenient to think about these as two separate families of architectures

medical images, cognitive scores, genetic factors etc. (see Section 2.3), change in signals in different locations in feature space (disparate brain locations or different cognitive tests) may correspond to disparate *pathological* signatures of the disease. Hence, we restrict to fully connected designs for the presentation of the proposed ideas – which nevertheless can be applied to convolutional networks as well.

Denoising Autoencoders (DA) and Stacked DA (SDA). An autoencoder is a single-layer network that learns robust *distributed representations* of the input data. Given inputs $\mathbf{x}_i$, the autoencoder learns hidden/latent representations $\mathbf{h}_i = \sigma(\mathbf{W}\mathbf{x}_i + b)$, such that the reconstructions $\hat{\mathbf{x}}_i = \sigma(\mathbf{W}^T\mathbf{h}_i + c)$ are as *close* as possible to $\mathbf{x}_i$. It minimizes the following input reconstruction error

$$\mathcal{Z}_a(\{\mathbf{x}_i\}_1^n, \theta) \coloneqq \underset{\mathbf{W},b,c}{\arg\min} \sum_{i=1}^{N} \mathcal{L}\left(\mathbf{x}_i, \sigma(\mathbf{W}^T\mathbf{h}_i + c)\right) \quad \text{s.t.} \quad \mathbf{h}_i = \sigma(\mathbf{W}\mathbf{x}_i + b) \tag{4.3}$$

where $\mathcal{L}(\cdot,\cdot)$ denotes a suitable loss function. Observe that with no other constraints on $(\mathbf{W}, b, c)$, the above minimization could potentially learn *identity* mappings, i.e., $\mathbf{h}_i$s will be identical to the inputs, making the autoencoding setup useless. Several approaches have been suggested to avoid such identity mappings and instead learn useful representations (Bengio, 2009b). We consider the approach where the inputs $\mathbf{x}_i$ are *corrupted* stochastically and the autoencoder is forced to reconstruct the original (non-corrupted) versions. This is referred to as a *denoising* autoencoder (DA) (Vincent et al., 2010), and the minimization in (4.3) will change to

$$\mathcal{Z}_{da}(\{\mathbf{x}_i\}_1^n, \theta) \coloneqq \underset{\mathbf{W},b,c}{\arg\min} \sum_{i=1}^{n} \mathbb{E}_{\tilde{\mathbf{x}}\sim\gamma(\tilde{\mathbf{x}}|\mathbf{x})}\mathcal{L}(\mathbf{x}_i, \sigma(\mathbf{W}^T\sigma(\mathbf{W}\tilde{\mathbf{x}}_i + b) + c)) \tag{4.4}$$

where $\gamma(\cdot)$ is a stochastic corruption function, and $\tilde{\mathbf{x}}_i$ represents the corrupted $\mathbf{x}_i$. $\gamma(x_{ij}) = x_{ij}$ with some (given) probability ζ and 0 elsewhere ($j = 1, \ldots, d$ are the data dimensions). DA is a stochastic autoencoder whose learning procedure seeks to undo the input corruptions, and hence the name, denoising. The corruption forces the transformations to correspond to some properties of input data, since the reconstruction error in (4.4) decreases only if the transformations pick out the most informative data dimensions. Hence $\mathbf{h}_i$s are abstract enough to generate the inputs (Vincent et al., 2010). Figure 4.2(b) summarizes the DA learning. Multiple DAs can be concatenated to construct a *stacked* DA (SDA), where the hidden representations of l^{th} DA are the uncorrupted inputs to $(\ell+1)^{th}$ DA ($\ell = 1, ldots, L-1$). The objective of SDA is

$$\mathcal{Z}_{sda}(\{\mathbf{x}_i\}_1^n, L, \theta) := \sum_{l=0}^{L-1} \mathcal{Z}_{da}(\{\mathbf{h}_i^l\}_1^n, \theta) \;\; ; \;\; \mathbf{h}_i^l = \sigma(\mathbf{W}^l \mathbf{h}_i^{l-1} + p^l) \; ; \; \mathbf{h}_i^0 = \mathbf{x}_i \quad (4.5)$$

It is straight forward to see that the structure of SDA is identical to that of a L-layered MLNN, and the learned parameters $(\mathbf{W}^\ell, b^\ell, c^\ell)$ can be used to initialize the MLNN. The final layer $\mathbf{h}_i^\ell$ can then be compared to the desired outputs and the errors can be propagated back to fine-tune the network. SDA learning proceeds layer-wise.

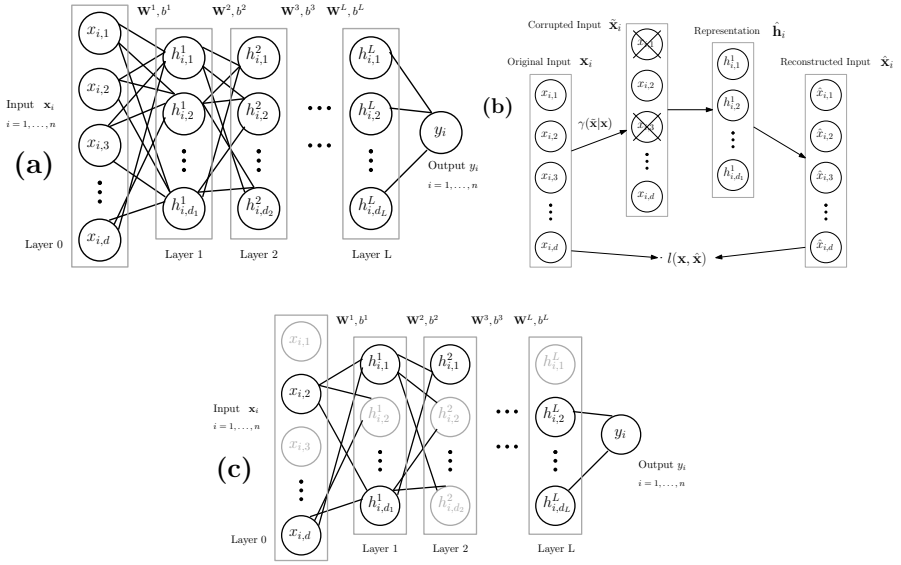


Figure 4.2: MLNN architecture and the learning process of DA and Dropout.

Dropout Networks Unlike SDAs where the corruption is applied stochastically in a layer-wise fashion, the dropout network simply drops a fraction of the network units across *all* layers. Figure 4.2(c) shows the dropout learning procedure. During the training stage, in each iteration of the back-propagation, the transformation parameters of a smaller sub-network are updated. The size of this sub-network is determined by the dropout rate ζ (which, in general, is the same across all layers). The sub-networks across different iterations are chosen randomly, and hence, each transformation parameter is updated approximately

$(1-\zeta)t$ times (where t is the number of gradient iterations). During the test time, the learned parameters are scaled up by $(1-\zeta)$ correspondingly to predict the final layer representation of the input. One can perform pretraining type initialization in tandem with dropout since the latter does not work layer-wise (Srivastava et al., 2014). In principle, both SDA and dropout are different types of feature corruption based regularization schemes, and as presented earlier, the dropout learning procedure has been introduced to address the overfitting issue in MLNNs (Srivastava et al., 2014). However, (Baldi and Sadowski, 2014; Krizhevsky et al., 2012) have shown that, in certain cases, pretraining can be 'by-passed' completely when using dropout. This is because the dropout dynamics result in smaller networks, there by reducing the effect of the local minima while also resulting in parameter estimates that are generalizable and robust to input perturbations. On top of the dropout regularization, it is typical to use the batch normalization as well when training the networks. With this background we now present the rationale behind our proposed deep learning models for the small sample regime followed by their structure and learning procedure.

4.3 Optimal enrichment criterion

The ideas driving our randomized deep networks are motivated directly from the design of optimal enrichment criterion. The sample size estimation from (4.1) suggests that, for a given significance α and power $1-\beta$, the required sample size s (per arm) *increases* as the standard deviation of the outcome σ *increases*, and/or, as the standard change in the outcome δ *decreases*. Recall that the hypothetical RCT presented in Section 4.2 asks for the change in outcome to decrease by $(1-\eta)$ (or the disease should reduce by η). Hence, a smaller η directly indicates that a smaller drug induced change in the treatment is desired and can be detected, there by increasing the robustness of the trial. Clearly, for a fixed number of subjects s, decreasing σ and/or increasing the mean change in the outcome δ, will decrease the detectable drug effect η. Hence, (4.1) implies that one can design an efficient clinical trial by selecting the population and the outcome such that, during the span of the trial, there will be large longitudinal changes in the outcome δ *and* small outcome variance σ^2. Ideally, the trial should *not* include subjects who remain healthy (and/or do not decline) as time progresses because they will reduce the trial's sensitivity for detecting any drug effect. Nevertheless, in general, the trial is always diluted by includ-

ing those subjects who are unlikely to benefit from the drug, indicating that the alternative hypothesis that the drug induces some effect is moot (for this subgroup). Removing such weak decliners from the trial will result in large δ, but may not necessarily ensure smaller σ^2 for the outcome. To address this, we need to *explicitly* reduce the outcome variance; this is generally not feasible because the outcomes are cognitive and/or blood flow based measures whose statistical characteristics (range, median, structure etc.) cannot be altered.

An alternative approach to ensure smaller outcome variance is by designing a *computational* disease marker that predicts the decline in the disease with high confidence. This new marker yields smaller variance. If the correlation of this computational marker with the intended trial outcome is strong, then the low variance characteristic of the new marker translates, to a certain extent, to the outcome due to the marker's strong predictability of future decline. Hence, once the new marker is established to have strong correlations to the outcome, one can use it as an "inclusion or enrichment criterion" to filter the trial population – those subjects whose future decline is small according to this enrichment criterion are removed from the trial. An optimal enrichment criterion would then be a computational disease marker with strongest predictability of the disease with smallest possible variance across subjects. Note that the intuition behind this is that reduction of this enricher's variance will indirectly reduce the outcome variance. Using an existing disease marker may not necessarily guarantee this behavior, and hence, one needs to explicitly design such a criterion. In summary, the optimal inclusion criterion should have the following properties.

- strong discrimination power for different stages of the disease – i.e., *no approximation/modelling bias*;
- strong correlation with an existing disease outcomes or other biomarkers – i.e., strong *predictive power* of the disease; and
- small prediction variance – i.e., *small variance* on the intended outcome.

The above requirements can be formulated as a statistical estimation problem. Given the inputs, which may include medical imaging data and/or other relevant types of clinical and/or demographic information, the estimator output is a *new* disease marker satisfying the above requirements. No approximation bias and strong predictive power implies that the estimator needs to be unbiased with respect to the classes/labels. Concurrently with the low prediction

variance, the problem of designing the optimal enrichment criterion reduces to constructing a *minimum variance unbiased (MVUB) estimator* of the disease.

Ensemble learning and Randomization. The existence of a MVUB estimator is governed by whether the Cramer-Rao lower bound can be achieved, i.e., any unbiased estimator that achieves this lower bound is referred to as a MVUB (Kay, 1993). However, finding such an unbiased estimator can be difficult, and especially, in the small-sample regime where $d \gg n$, computing the lower bound itself is problematic. Instead, in such settings an alternative approach to designing MVUBs is by first generating sets of unbiased estimators that are approximately *uncorrelated* to each other (in the ideal case, independent), and combine them in some reasonable manner to reduce the variance while retaining unbiasedness. This is the classical bootstrap approach to MVUB design in high-dimensional statistics (Smith, 2005), and the family of models that adapts this approach is broadly referred to as "ensemble learning" methods (Dietterich, 2000). The variance reduction behavior follows from ensuring that the estimators/learners have sufficiently small cross-correlation, and so, their linear combination will have smaller variance compared to that of each individual estimator/learner. To see this, let $\mathbf{z}_k$ for $k = 1, \ldots, K$ denote K different random variables (e.g., the outputs from K different estimators/learners), and $\bar{\mathbf{z}}$ denote their mean. Assuming, without any loss of generality, that the variance and cross-correlation of $\mathbf{z}_k$s to be σ^2 and ρ respectively, we have $\mathrm{Var}(\mathbf{z}_k) = \sigma^2$, $\mathrm{Cov}(\mathbf{z}_k, \mathbf{z}_{k'}) = \rho$, and $\mathrm{Var}(\bar{\mathbf{z}}) = \frac{\sigma^2}{K} + \frac{(K-1)\rho\sigma^2}{K}$

Depending on ρ, the variance of $\bar{\mathbf{z}}$ goes from $\frac{\sigma^2}{K}$ to σ^2 (under the assumption that $\rho > 0$ i.e., the estimators are not negatively correlated). In the current setting, since all the K estimators/learners share the same set of input data, they cannot be independent. However, by ensuring that ρ is as small as possible and increasing K, one can make sure that the composed estimator $\bar{\mathbf{z}}$ will have small variance. Several approaches may be used to ensure small ρ, most of which are based on *randomly* dividing the input dimensions, data instances and/or other estimation/learning parameters into K subsets and constructing one estimator from each of these subsets (Dietterich, 2000). The outputs of these estimators can then be considered to be random (or stochastic, in some sense) approximations of the ideal output with zero-bias and small variance. There is extensive empirical evidence for such strategies where the eventual *ensemble learner* will retain the discriminative power of the individual weak learners while reducing the apparent prediction variance (Dietterich, 2000). We

will use deep networks to construct an efficient weak learner, followed by a systematic strategy to construct the ensemble – this overall learning procedure will correspond to the randomized deep network model.

4.4 Randomized Deep Networks

The ideas from ensemble learning do address the variance reduction requirement for the optimal enrichment criterion. However, the weak learners that go into this ensemble need to be unbiased to begin with. Recall the success of deep learning in learning complex concepts in computer vision, natural language processing and information retrieval (Kavukcuoglu et al., 2010; Krizhevsky et al., 2012; Dahl et al., 2011). It is reasonable to expect that these methods should be translatable to learning problems in medical imaging and neuroimaging with improved performance than the state-of-the-art. We will setup the desired ensemble MVUB using deep network weak learners that are trained appropriately to predict the disease. There are a few caveats, however, as described below.

Small Sample Regime and Multiple Modalities

Although the pretraining idea in tandem with the dropout learning addresses the issue of non-convexity to a certain extent, (Erhan et al., 2009; Bengio, 2009b; Ngiam et al., 2011) have given evidence that one of the main reasons for the success of deep learning is the availability of large number of unsupervised and/or supervised training instances. The few studies that apply deep learning methods in neuroimaging have reported such observations as well (Gupta et al., 2013; Plis et al., 2013). Simply stated, the non-convexity, together with the stochasticity that comes from the corruption in DA or the dropout process, demand a very large number of gradient search iterations and data instances to effectively search through the solution space in computing generalizable solutions. As the data dimensionality increases, the dataset size required to ensure that sufficiently many combinations of corrupted/dropped dimensions are available also increases proportionally. But the fundamental difference between vision applications, and medical imaging and bioinformatics is the lack of such large datasets. In machine vision, one has access to extremely large datasets (on the orders of millions of images) – including both large number of unlabeled and supervised instances (e.g., in object recognition, document analysis and so on). On the other hand, a typical voxel wise imaging study, for

instance, will have $n < 500$ subjects while the number of voxels/features (d) will exceed a million – the classical small-sample regime.

Further, in contrast to classical machine learning tasks, the problems in bioinformatics involve data from multiple acquisition types/domains (e.g., brain imaging data including Magnetic Resonance images (MRI), Positron Emission Tomographic (PET) images, several types of cognitive and neuropsychological scores, vascular and blood perfusion data, genetic single nucleotide polymorphisms). Most applications in these areas would require efficient statistical models for 'fusing' such multi-modal data, especially because they provide distinctive information about the underlying disease. Such multi-modal studies have always been shown to result in higher performance than uni-modal ones (Hinrichs et al., 2011b). Using classical versions of deep architectures, including SDAs or dropout networks, for these multi-modal problems may result in unreliable outputs, with no guarantee of generating either stable or generalizable solutions. In part this follows from the fact that the minimum number of training instances cannot be smaller than a (typically unknown) Nyquist threshold for efficient estimation of the underlying concepts (Bishop, 2006).

One of the main contributions of our proposed randomized deep networks is to translate the success of deep learning to multi-modal neuroimaging problems in the small sample regime. The simplest solution to mitigating the $d \gg n$ issue is by pre-selecting the most influential voxels using some statistical test (e.g., t-test based on some known grouping information) and using only this subset as features within the learning framework downstream. However, this proposal is lossy, in the sense that, non-selected features are discarded irrespective of how discriminative they are.This may mean that the performance will be driven by the 'goodness' of pre-processing. (Bourgon et al., 2010) discuss this need for avoiding bias and influence of data processing on the false positive error rates for the eventual learning task. On the other hand, one can avoid the feature screening completely by working with slices of the input data (e.g., 2D slices or smaller resolution images from a 3D image). Although this is lossless, working with one slice at-a-time will restrict interactions of voxels and brain regions from anatomically far apart regions. But all brain regions may have complex biological interactions in generating the final label (e.g., the disease status). An alternative to these extremes is to *categorize* (or *tessellate*) the entire set of voxels into multiple subsets (e.g., spatially contiguous blocks) and learn a network (e.g., SDA or dropout network) on each block separately while allowing for

different blocks to interact with each other in some prescribed manner. The network learned in each block will then serve as a weak learner in the above described ensemble (as in boosting (Dietterich, 2000)). Later, the individual block-wise networks, which interact with each other, can be combined in a meaningful manner yielding a better fit for the dependent variable $\mathbf{y}$.

Roadmap

We will show that the above described proposal is viable. Observe that if the number of voxels in each subset/block is comparable to n, and much smaller than the input dimensionality allowed in this smaller subset (denoted by $\tilde{d}$), the learning problem at the level of *this* block is well defined. For instance, d voxels can be divided into B subsets and a SDA or dropout network can be trained on each block separately. On its own, each block's output will not be useful since it corresponds to only a small number of features. But our final output will utilize the outputs from all B blocks, treating each as a stub (i.e, weak learner). This scenario assumes that the tessellation (i.e., the voxel to block mapping) is given or fixed. But in practice, the "correct" tessellation is not known in advance. However, it is possible to marginalize over this as described below. Consider the set of all possible tessellations $\mathcal{C}$ (an exponentially large set). Constructing a learning model, however simple, for each item in $\mathcal{C}$ is unrealistic. Instead, we can use the standard trick of marginalizing over $\mathcal{C}$ by drawing a large number of samples from it to approximate the summand. In other words, by resampling from $\mathcal{C}$ and deriving many possible B (sufficiently large) number of subsets (and learning network weights for them), we can obtain partial invariance to the lack of a correct tessellation. Note that one realization of this process corresponds to drawing a sample from $\mathcal{C}$; this process provides *randomization* over clusterings where B different predictions from input instance are combined to generate a single classification/regression output.

The number of blocks B can be fixed ahead of time. The process of assigning voxels to blocks can leverage domain information like neighborhood interactions and/or consistency of correlations across all d dimensions or any other randomized procedure. For instance, if it is known that a local neighborhood has strong interactions, then each such neighborhood can constitute a single block. To allow for arbitrary brain regions to interact with each other, possibly within a block, the block construction should group arbitrary voxels together. Hence, to balance this arbitrary voxel selection and domain information driven

block generation, we first 'rank' all the voxels according to some information criterion (e.g., Kulback-Liebler divergence, Entropy). We then sample the voxels (for a given block) without replacement according to the cumulative distribution of these ranks. If these blocks are *sufficiently* independent, then any linear combination of their outputs will have smaller variance resulting in an MVUB as desired. However, the input data to all the blocks comes from the same subject, implying that the $\mathtt{B}$ weak learners will always be correlated. Nevertheless, we can force them to be approximately uncorrelated by adding another level of randomization over the block generation process. To do this, observe that, beyond the block generation, there are other sets of hyper-parameters which correspond to the learning mechanism of individual blocks like denoising rate, dropout rate, gradient stepsizes etc. Hence, for a given block we can learn $\mathtt{T}$ different number of learning models by randomly generating $\mathtt{T}$ different sets of such learning hyper-parameters.

This two-fold randomization will result in an ensemble of $\mathtt{B} \times \mathtt{T}$ weak learners with small correlation among them as possible. Clearly, increasing the number of weak learners by $\mathtt{T} > 1$, decreases the variance of MVUB, while also mitigating the influence of learning parameters. This is the idea of our randomized deep networks. Specific details about the randomization process will be described in the following section as we present the architecture and training mechanisms for these models. Lastly, observe that it is easy to incorporate multi-modal features like MRI or PET within our setup. The simplest way is to generate blocks from each of the modalities independently, train them separately, and combine their outputs at a later stage. If $\mathtt{m}$ denotes the number of modalities, then the blocks from each modality can simply be concatenated resulting in $\mathtt{mB}$ number of weak learners. Alternatively, one can construct 'cross-modal' blocks where voxels across multiple modalities are sampled and assigned to a single block. This procedure will have to take into account the modality specific tessellations and is more complex than the intra-modal design mentioned earlier. Since the networks are trained locally on each block, once the blocks are fixed, the voxels within a block do not *directly* interact with those from the other. One can nevertheless construct a feedback procedure that reassigns voxels among blocks based on the goodness of predictions from the previous set of blocks. This feedback is computationally expensive and it may not improve the performance whenever the number and size of the blocks are large. We refer to the two proposed randomized deep network models as – randomized Denoising Autoencoders (rDA) and randomized Dropout

Networks (rDr).

rDA and rDr training

Let $\mathcal{V} = \{v_1, \cdots, v_d\}$ denote the set of voxels. Consider a probability distribution $\tau(v)$ over the voxels $v \in \mathcal{V}$. This is the sampling distribution that governs the block construction i.e., the assignment of voxels to blocks, and in the simplest case, it is a uniform distribution. For each block $b = 1, \ldots, B$, we sample $d_b \ll d$ (fixed a priori) number of voxels without replacement using the distribution $\tau(v)$. We call this set of voxels, s_b. Each block is presented with T different sets of learning parameters (i.e., denoising rate, gradient learning rate and so on) denoted by $\theta_t \in \Theta$ for $t = 1, \ldots, T$, where Θ is the given hyper–parameter space. This means that each sample from the hyper-parameter space yields one weak learner i.e., one SDA or Dropout network for one block. Hence, a total of $B \times T$ number of weak learners are constructed. If $\tau(\cdot)$ is uniform, then asymptotically we expect to see one voxel in at least one of the B blocks. As discussed earlier, instead of uniform $\tau(\cdot)$, alternative choices may be used depending on prior information about the importance of including a particular voxel in one/more blocks. Depending on $\tau(\cdot)$ the blocks may be mutually exclusive. The influence of model hyper parameters including B, T, the number of voxels per block d_b, the sampling distribution $\tau(\cdot)$, and the robustness of the model to these choices are described in Section 4.4.

Figure 4.3 shows the architecture of randomized deep network with B blocks, each with T weak learners, where each weak learner corresponds to a L-layered MLNN (see Figure 4.3). The outputs from the ensemble of $B \times T$ networks are combined using ridge regression. Algorithm 4.4 summarizes the training procedure. Given training data $(\mathbf{x}_i, \mathbf{y}_i)$ for $i \in \{1, \ldots, n\}$, we first learn the unknown transformations – $(\mathbf{W}^l_{b,t}, p^l_{b,t}, q^l_{b,t}), \forall b \in \{1, \ldots, B\}; t \in \{1, \ldots, T\}; l \in \{1, \ldots, L\}$. Depending on whether the weak learner is an SDA or a dropout network, the learning process for these $B \times T \times L$ transformations will follow the minimization of (4.4) and (4.5), or the discussion from (4.2) respectively. The $\mathtt{Reweigh}(\cdot)$ operation in Algorithm 4.4 skews the sampling distribution to ensure that the un-sampled voxels (i.e., voxels that are not assigned to any block yet), will be given priority in the later blocks, while avoiding oversampling of the same set of voxels across multiple blocks.

Concatenating the L^{th} layer outputs from $B \times T$ learners, we get $\mathbf{H}_i = [[\mathbf{h}^L_{b,t}]]^{B,T}_{1,1}$. The weighted regression pooling then composes these outputs.

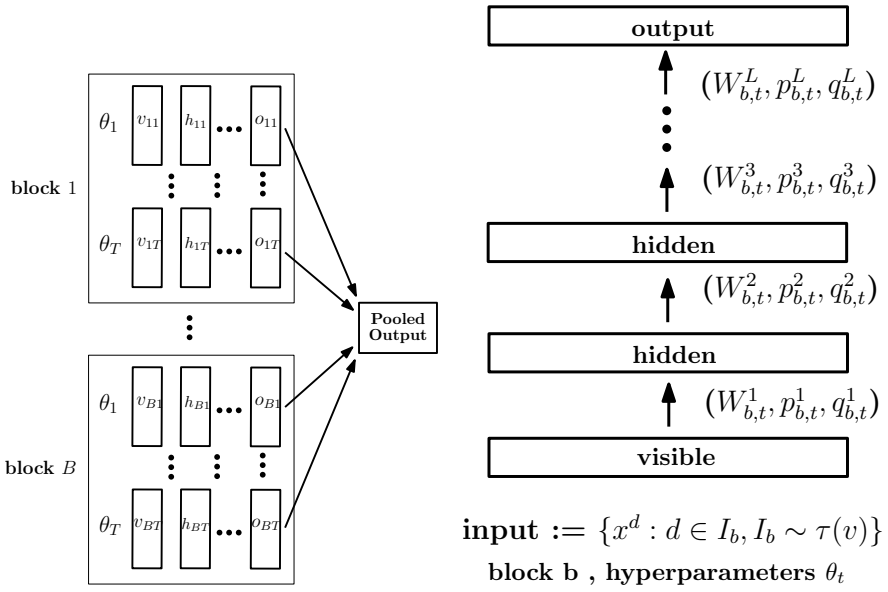


Figure 4.3: The architecture of randomized deep network.

$$\mathbf{U} \leftarrow (\mathbf{H}^{\mathsf{T}}\mathbf{H} + \lambda\mathbf{I})^{-1}\mathbf{H}^{\mathsf{T}}\mathbf{Y} \quad ; \quad \mathbf{H} = [[\mathbf{H}_i]]_1^n \quad ; \quad \mathbf{Y} = \left[\left[\mathbf{y}_i\right]\right]_1^n, \tag{4.6}$$

where $\mathbf{U}$ are the regression coefficients, λ is the regularization constant and $\mathbf{Y} = [\mathbf{y}_i]_1^n$. Since the networks are already capable of learning complex concepts (see discussion from Section 4.2), the pooling operations that we used was a simple linear combination of the $B \times T$ outputs with the ℓ_2-loss. Clearly, instead of regression, a simple mean of the outputs may also suffice to generate the final output because the networks are already ensured to be as uncorrelated as possible, and so, their mean output is a reasonable estimate of the predicted labels. Observe that the training algorithm, and the design of randomized deep networks in general, are agnostic to the type of architectures that were used as weak learners. Once the randomized network is trained, its prediction on a new test instance/example is given by (the scaling up of transformations in dropout network case are not shown here to avoid clutter, see (Srivastava et al., 2014) for more details),

$$\hat{\mathbf{y}} = \mathbf{h}\mathbf{U} \quad ; \quad \mathbf{h} = [[\mathbf{z}_{b,t}\mathbf{h}^{L}_{b,t}]]^{B,T}_{1,1} \; ; \; \mathbf{h}^{l}_{b,t} = \sigma(\mathbf{W}^{l}_{b,t}\mathbf{h}^{l-1}_{b,t} + \mathbf{p}^{l}_{b,t}) \; ; \; \mathbf{h}^{0}_{b,t} = \mathbf{x}, \quad (4.7)$$

Algorithm *Randomized deep networks – blocks training*

Input: $\theta_t \sim \Theta, \mathcal{V}, B, s_B, L, T, \mathcal{D} \sim \{\mathbf{x}_i, \mathbf{y}_i\}_1^n, \lambda$
Output: $(\mathbf{W}^{l}_{b,t}, \mathbf{p}^{l}_{b,t}, \mathbf{q}^{l}_{b,t})$
  for $b = 1, \ldots, B$ **do**
    $I_b \sim \tau(\cdot)$
    for $t = 1, \ldots, T$ **do**
      $(\mathbf{W}^{l}_{b,l}, \mathbf{p}^{l}_{b,l}, \mathbf{q}^{l}_{b,l}) \leftarrow \mathcal{Z}.(\mathcal{D}, L, I_b, \theta_t)$
    end for
    $\tau(\mathcal{V}) \leftarrow \mathtt{Reweigh}(\tau(\mathcal{V}), I_b)$
  end for

Hyperparameters

The hyper parameters of randomized deep network include:

- B: The number of blocks.
- T: The number of hyper-parameter sets; depth of the network, gradient learning rate, denoising (for rDA) or dropout (for rDr) rates (or other appropriate regularization criteria depending on the weak learning mechanism), number of gradient iterations etc.
- $\tau(\cdot)$: The sampling distribution over all the voxels for constructing the blocks,
- d_b: The number of voxels within each block (or length of the input layer for weak learners).
- λ: The regularization parameter for ridge regression.

Observe that, within each block, the randomization is over the T set of learning parameters, and hence if the hyper-parameter space is sampled uniformly, the model's outputs will be robust to changes in T. The simplest choice for the block-wise sampler $\tau(\cdot)$ assigns uniform probability over all dimensions/voxels as described above. However, we can assign large weights on local neighborhoods which are more sensitive to disease progression, if desired. We can also setup $\tau(\cdot)$ based on entropy or the result of a hypothesis test. More precisely, entropic measures (like Kullback-Leibler divergence), t-scores or z-scores can be used to estimate the discrimination power of each voxel (this would correspond to performing V number of hypothesis tests; uncorrected). The resulting

scores can then be normalized and used as the sampling distribution $\tau(\cdot)$. The $\mathit{Reweigh}(\cdot)$ step in Algorithm 4.4 takes care of previously unsampled dimensions/voxels. The simplest such re-weighting includes removing the already sampled voxels from $\tau(\cdot)$ (which is the same as sampling voxels without replacement). Although there is no analytic way to setup B and d_b (for $b = 1, \ldots, B$), a reasonably large number of blocks with $d_b = d/B$ would suffice (refer to discussion in Section 4.5 about the choices made in our experiments). The influence of dropout and denoising rates on the performance of deep networks have been well studied empirically (Vincent et al., 2010; Srivastava et al., 2014), and (Baldi and Sadowski, 2014) analyze the dynamics of the dropout networks as the dropout rate changes. Since such studies already provide ample evidence for setting these rates, we do not explicitly analyze them for our setting.

The disease markers – rDAm and rDrm

The randomized deep network from Figure 4.3 and Algorithm 4.4 will now be adapted to the problem of designing a MVUB of disease spectrum. Depending on the choice of architectures used, these MVUBs will be referred to as randomized Denoising Autoencoder marker (rDAm) or randomized Dropout network marker (rDrm). The sigmoid non-linearity ensures that the outputs of individual blocks lie in $[0, 1]$, and so the predictions from the rDA and rDr models on new test examples are bounded between 0 and 1. This is clearly advantageous since a bounded predictor would implicitly guarantee bounded variance (a desirable property from the perspective of MVUB design) while also covering the entire disease spectrum. Hence, we only need to ensure that each individual block is an unbiased estimate of the disease. We then train the weak learners (the blocks) *only* using healthy controls and completely diseased subjects each labeled $y = 1$ and $y = 0$ respectively. Here the diseased subjects would be those with clinical AD, and all subjects in other stages of the disease (like early or late MCI (Jelic et al., 2006; Petersen, 2007; Aisen, 2011)) are not going to be used for training. Since the pooling operation corresponds to a regression (see (4.7)), the test time predictions are the desired disease markers rDAm or rDrm. They will correspond to the *confidence* of rDA or rDr in predicting the subject's decline – closer to 1 if the subject is healthy or 0 is not. Clearly, changing the training setup from regression to classification will modify the interpretation of these predicted markers, but their properties like bounded variance and MVUB will still remain the same.

Recall the discussion in Section 4.2 about the sample enrichment procedure where the inclusion criterion decides whether the subject needs to be enrolled in the trial or not. As noted in Section 4.1, from the practical perspective, the inclusion criterion should filter subjects at the trial start point itself. Specifically, the inclusion criterion, either rDAm or rDrm computed at the *baseline* (or trial start point) will then be used to enroll the subjects. Since the markers are bounded between 0 and 1, the enrichment is driven by choosing the "appropriate" *threshold* or cut-off to retain subjects accordingly. This procedure is summarized here:

- The first check for performing this baseline sample enrichment is to ensure that these markers, computed at the baseline, have strong correlation (or dependance) with other disease markers, some of which would indeed be the intended trial outcomes (Jack et al., 2013; Weiner et al., 2013). If the dependencies turn out to be significant, this is evidence of convergent validity, and using baseline rDAm or rDrm as inclusion criteria for enriching the trial population is, at minimum, meaningful.
- Once this is the case, using the enrichment threshold $\mathrm{t}(0 < \mathrm{t} < 1)$, the enriched cohort would include only those subjects whose baseline rDAm or rDrm is smaller than t (closer to being diseased). Alternatively, by avoiding to choose the optimal cut-off t, one can instead include a fixed fraction (e.g., $1/4$th or $1/3$rd) of the whole population whose baseline rDAm or rDrm is closest to 0 (fixing a fraction automatically fixes t).

Clearly, the choice of t is vital here, and one way to choose it is by comparing the mean longitudinal change of some disease markers (MMSE, CDR and so on) for the enriched cohort as t goes from 0 to 1. The optimal t would correspond to a discontinuity or a bump in the change trends as t increases. We discuss these issues when we evaluate rDA and rDr shortly.

4.5 Experiments

Participant Data and Preprocessing

Imaging data including AV45 singular uptake value ratios (SUVR), FDG PET SUVRs and gray matter tissue probability maps derived from T1-weighted MRI data, and several neuropsychological measures and CSF values from 516 individuals enrolled in ADNI 2 were used in our evaluations. Of these 516

subjects (age 72.46 ± 6.8, female 38%), 101 were classified as AD (age 75.5 ± 5.1), 148 as healthy controls (age 70.75 ± 7), and 131 and 136 as early and late MCI (age 74.3 ± 7.1 and 75.9 ± 7.7) respectively at baseline. There was a significant age difference across the four groups with $\mathsf{F} > 10$ and $\mathsf{p} < 0.001$. Among the MCIs, 174 had positive family history (FH) for dementia, and 141 had at least one Apolipoprotein E (APOE) e4 allele. CSF measures were only available at baseline, and three time point data (baseline, 12 months and 24 months) was used for the rest. Modulated gray matter tissue probability maps were segmented from the T1-weighted MRI images (other tissue maps are not used in our experiments) using Voxel-based Morphometry (Ashburner, 2010). The segmented map was then normalized to Montreal Neurological Institute (MNI) space, smoothed using 8mm Gaussian kernel, and the resulting map was thresholded at 0.25 to compute the final grey matter image. All PET images were first co-registered to the corresponding T1 images, and then normalized to the MNI space. Manually constructed masks of pons, vermis and cerebellum were then used to scale these PET maps by the average intensities in pons and vermis (FDG PET SUVR) and cerebellum (florbetapir PET SUVR). All preprocessing was done in SPM8 (Ashburner et al.).

Evaluations Setup

We train the randomized deep networks using only baseline imaging data from all the three modalities, MRI, FDG PET, and AV45 PET with diseased (AD, labeled 0) and healthy (CN, cognitively normal, labeled 1) subjects. When testing on MCI subjects, these trained models output a multi-modal rDAm and rDrm, which represent the confidence of rDA and rDr that a given MCI subject is (or is not) likely to decline. We only use baseline imaging data for training, thereby making the models deployable in practice, while the predictions can be performed on MCIs at baseline and/or future time-points. $\mathsf{B} = 5000$, $\mathsf{d_b} = \mathsf{d}/\mathsf{B}$ (i.e., each voxel appears in only one block) and $\lambda = 1$ for both rDA and rDr. Here, $\tau(\cdot)$ is based on differentiating ADs and CNs using KL divergence (refer to Section 4.4). Multiple combinations of B, $\mathsf{d_b}$ and $\tau(\cdot)$ (including uniform and t-score based) were also evaluated, however, none of them gave any significant improvements over the above settings. The blocks construction for multi-modal rDA and rDr did not use cross-modal sampling to ensure manageable computational burden (see Section 4.4). Within this setup, our evaluations are three-fold. Since the test data are MCIs, we first

evaluate if baseline rDAm and rDrm differentiate early MCI from late MCI, and parental family history as a contributing risk factor. These evaluations include the baseline markers derived from seven different combinations corresponding to the three imaging modalities available.

After checking the construct that multi-modal markers are superior to uni-modal markers, we evaluate the premise whether the multi-modal rDA and rDr markers are good disease progression markers. We demonstrate this by computing the dependence of these multi-modal baseline rDAm and rDrm with well-known outcome measures including several cognitive and neuropsych scores as well as risk factors – see Section 2.3 from Chapter 2 for details. These outcomes are MMSE, ADAS Cog-13, MOCA, RAVLT, PsyMEM, PsyEF, hippocampal volume from grey matter tissues in MRI, CDR-SB, a binary marker for conversion from MCI to AD (DxConv), CSF levels including Tau τ, Phospho-Tau $p\tau$, Amyloid Beta $A\beta 42$, ratios of τ and $p\tau$ with $A\beta 42$, APOE allelle4 and maternal/paternal FH. For continuous markers, we used the Spearman Rank Order Correlation coefficient to assess the dependencies and accepted those statistics as significant whenever the corresponding $p < 0.05$. For binary markers, a t-test was used with same significance value. Observe that we are interested in evaluating the predictive power of baseline rDAm and rDrm. Specifically, we report the correlations of baseline rDAm with these markers at 12-months and 24-months, and also the longitudinal change in one year, thereby, providing evidence that whenever the baseline markers are closer to 0, the subject's longitudinal changes are in fact steeper/stronger.

Once the correlation construct is appropriately validated, we evaluate the use of baseline rDAm and rDrm for sample enrichment. We compute the sample sizes (using (4.1) based on the discussion in Section 4.2) required when using the above cognitive, neuropsychological, diagnostic and other imaging-based outcome measures with (and without) rDAm or rDrm based enrichment. The sample size trends are computed across different enrichment thresholds (see Section 4.4). For better interpretation of the estimates from the perspective of a practitioner or clinician, we estimate the effect size as a function of the marker enrichment cut-off for a given (fixed) sample size. We also compute the performance improvement from using our markers relative to the alternative imaging-derived enrichers including ROI summaries from FDG and florbetapir images [3] with particular attention to the current state-of-the-art imaging based

[3] FDG ROIs include Left Angular Lobe, Right Angular Lobe, Left Temporal Lobe, Right Temporal Lobe and Cingulate. AV45 ROIs include Frontal Lobe, Temporal Lobe, Parietal Lobe and Cingulate gray matter. The corresponding ROI measures are summed

computational summary measure that we refer to as a Multi Kernel Learning marker (Hinrichs et al., 2011a). MKLm is based on a Multi Kernel support vector machine (MKL) (Hinrichs et al., 2011a) that tries to harmonize contributions from multiple imaging modalities for deriving a maximum margin classifier in the concatenated Hilbert spaces. Unlike traditional support vector machines, MKLm uses a linear combination of kernels and solves for both the weights on the kernels as well as the normal to the hyper-plane concurrently (Hinrichs et al., 2011a). Similar to the proposed models, MKL is trained using AD and CN subjects, and the corresponding predictions on MCIs are referred to as MKL markers (MKLm). Note that whenever the labels correspond to continuous predictors instead of class indices (like AD vs. CN), we use ϵ-support vector machine version of MKL.

Results

Table 4.1 shows the discrimination power of rDA and rDr for classifying early and late MCI (Table 4.1(a)) and Family History (Table 4.1(b)). Both models perform better than the baseline MKL. Table 4.2 and 4.3 correspond to the predictive power of baseline rDAm and rDrm respectively. They show the Spearman correlations and t-statistics of the baseline multi-modal markers with cross-sectional scores (baseline, 12 and 24 months) and longitudinal change (12 and 24 months) in other disease markers. Negative correlations indicate that the corresponding measures (ADAS, τ, $p\tau$, $\tau/A\beta 42$, $p\tau/A\beta 42$) increase with disease progression. Across both rDAm and rDrm, large correlations with $r > 0.45$ and/or $p < 10{-}4$ (denoted by the superscript *), were observed with baseline summary measures (see the second column in Tables 4.2 and 4.3), specifically with ADAS, neuropsychological (memory and executive function) composite scores, hippocampal volume and CSF levels involving $A\beta 42$. For both rDA and rDr, FH had a smaller influence on baseline rDAm compared to APOE. All the cross-sectional correlations (columns $2-4$, Table 4.2 and 4.3) were significant ($p < 10^{-4}$). The correlations of baseline markers with longitudinal change (last two columns) were significant with $r > 0.21$ and $p < 0.001$ for all the measures (except few cases involving PsyEF and MOCA). rDAm's correlations are stronger than that of rDrm's across all the constructs. Tables 4.1–4.3 provide strong evidence for both rDAm's and rDrm's disease predictive power.

up to obtain single global summary for each of FDG and AV45.

Table 4.4, Figures 4.4 and 4.5 show the relevance of rDA and rDr for enrichment. Table 4.4 shows the coefficient of variation (CV, the ratio of standard deviation to mean) of rDAm and rDrm for three different populations of interest – all MCIs, late MCIs and MCIs with positive FH. The proposed markers' CV is smaller than MKLm for all three populations and combinations of modalities – making it a better candidate to be used as a prediction measure as well as an enricher (refer to (4.1), where the right hand side includes terms depending on CV^2). CV of rDAm is smaller than rDrm. Each plot in Figures 4.4 and 4.5 correspond to the mean longitudinal change of some disease marker, *after* the MCI population (the test set) is enriched by removing the weak decliners (subjects with baseline rDAm or rDrm above a certain cut-off t, shown on the x-axis). For both rDA and rDr, the plots show that MMSE, MOCA, Hippocampal Volume, CDR-SB and DxConv have large changes when weak decliners are progressively removed – strong evidence for rDAm's and rDrm's predictive power. Specifically, for some measures the changes are much steeper for 24 months than 12 months (black and blue lines in each plot). PsyEF resulted in irregular changes at different time points for both rDA and rDr.

Tables 4.5-4.8 show sample sizes when multi-modal baseline rDAm and rDrm are used as enrichers, at 80% statistical power and 0.05 significance level, with a hypothesized treatment effect of $\eta = 0.25$ (i.e., 25% decrease in the disease). Tables 4.5 and 4.6 show the sample sizes at four different rDAm and rDrm enrichment cut-offs (third to last columns) respectively. Recall that higher values of the markers at baseline imply closer to being healthy. Hence, enrichment entails filtering out all subjects whose baseline rDAm or rDrm are above some chosen cut-off. Results show that, compared to the no-enrichment regime (second column), the sample estimates from rDAm or rDrm enrichment are significantly smaller, with more than 5 times reduction when using bottom 20% and 25% percentiles (third and fourth columns). The sample sizes from rDAm enrichment are smaller than those from rDrm, following the previous observation that the CV's of rDrm are larger (refer to Table 4.4). In particular, with rDAm; MMSE, CDR-SB and DxConv give consistently smaller estimates (200 to 600) across all columns (the four different percentiles). ADAS and PsychEF still required very large sizes (774 and > 2000 respectively) even at 20% enrichment percentile. Similar trends are observed for rDrm.

Tables 4.5 and 4.6 further enrich the population after using rDrm and rDrm with extra covariate information like FH and/or APOE status. Specifically, here we explicitly filter out those MCI subjects who are *not* FH and/or APOE

positive before performing baseline rDAm or rDrm enrichment. Clearly, the sample sizes further decrease because of this extra filtration as shown in last three columns of Tables 4.5 and 4.6. APOE as a covariate resulted in smallest possible estimates in general (< 350 per arm with MMSE, CDR-SB and DxConv outcomes) across all the outcomes except PsyEF (last two columns in Table 3). Interestingly, they are smaller than the sample sizes from using both APOE and FH as covariates (last column). DxConv as an outcome with rDAm or rDrm + APOE enrichment yields a sample size of 170 and 219 respectively. Figure 4.6 and 4.7 shows the detectable effect sizes as rDAm and rDrm enrichment cut-offs are varied for a fixed sample size of 500 per arm. Observe that the detectable effect size ($1 - \eta$) decreases as more weak decliners are filtered out. This can be seen by the increase of η (y-axis) as the rDAm or rDrm cut-offs on x-axis decrease, specifically for MMSE, CDR-SB and DxConv outcomes. These plots are useful from a clinician's (practitioner's) perspective, as will be discussed later in Section 4.6. Finally, Table 4.9 compares our proposed enrichers with other imaging-derived inclusion criteria (the cut-off for all the enrichers corresponds to including the strongest 20% decliners in their respective scales). Both rDAm and rDrm consistently outperformed other alternatives (and between them rDAm was better), with up to 2 times smaller estimates than MKLm, and much larger reductions compared to uni-modal summaries (hippocampal volume, FDG and AV45 ROIs).

Table 4.1: rDA and rDr vs. MKLm (F-statistics). A : AV45, F : FDG and T : T1GM

Model	Amyloid	FDG	T1GM	A+F	A+T	F+T	A+F+T
(a) Early versus Late MCI							
MKL	$20.5^{\dagger}$	$16.8^{\dagger}$	$16.5^{\dagger}$	$16.4^{\dagger}$	$20.4^{\dagger}$	23.6^{*}	27.9^{*}
rDA	22.1^{*}	$9.7^{\dagger}$	$20.0^{\dagger}$	$19.5^{\dagger}$	24.1^{*}	21.2^{*}	27.6^{*}
rDr	$20.1^{\dagger}$	$11.5^{\dagger}$	$20.3^{\dagger}$	$17.5^{\dagger}$	23.0^{*}	21.2^{*}	26.9^{*}
(b) Family History Positive versus Negative							
MKL	4.3^{**}	$7.5^{\dagger}$	5.3^{**}	$7.3^{\ddagger}$	$6.8^{\ddagger}$	6.6^{**}	$8.3^{\ddagger}$
rDA	4.7^{**}	$11.8^{\dagger}$	$11.2^{\dagger}$	$6.8^{\dagger}$	$12.4^{\dagger}$	$13.2^{\dagger}$	$13.3^{\dagger}$
rDA	4.6^{**}	$9.9^{\dagger}$	$12.0^{\dagger}$	$6.9^{\dagger}$	$11.7^{\dagger}$	$13.2^{\dagger}$	$12.0^{\dagger}$

4.6 Discussion

The ability to design clinical trials with smaller sample sizes but sufficient statistical power will enable the implementation of affordable, tractable and hopefully, conclusive trials. Efficiency is seriously compromised in trials where

Table 4.2: Correlations of multi-modal baseline rDAm

Biomarker	Baseline	Cross-Sectional		Longitudinal Change	
MMSE	0.39^{*}	0.49^{*}	0.45^{*}	$0.21^{\dagger}$	$0.33^{\ddagger}$
ADAS	-0.56^{*}	-0.58^{*}	-0.53^{*}	$0.21^{\dagger}$	$-0.53^{\ddagger}$
MOCA	0.48^{*}	0.51^{*}	0.59^{*}	0.06	0.59^{*}
RAVLT	0.49^{*}	0.52^{*}	0.57^{*}	0.13^{**}	$0.57^{\dagger}$
PsyMEM	0.56^{*}	0.57^{*}	0.59^{*}	0.28^{*}	$0.42^{\dagger}$
PsyEF	0.52^{*}	0.57^{*}	0.46^{*}	0.15^{**}	0.26^{**}
HippoVol	0.72^{*}	0.74^{*}	0.79^{*}	0.33^{*}	0.47^{*}
CDR-SB	-0.33^{*}	-0.49^{*}	-0.55^{*}	-0.36^{*}	-0.53^{*}
DxConv	$--$	21^{*}	31^{*}	21^{*}	31^{*}
τ	-0.39^{*}	$--$	$--$	$--$	$--$
$p\tau$	-0.40^{*}	$--$	$--$	$--$	$--$
$A\beta42$	0.55^{*}	$--$	$--$	$--$	$--$
$\tau/A\beta42$	-0.52^{*}	$--$	$--$	$--$	$--$
$p\tau/A\beta42$	-0.52^{*}	$--$	$--$	$--$	$--$
APOE	$3.47^{\dagger}$	$--$	$--$	$--$	$--$
FH	2.16^{**}	$--$	$--$	$--$	$--$

Table 4.3: Correlations of multi-modal baseline rDrm

Biomarker	Baseline	Cross-Sectional		Longitudinal Change	
MMSE	0.37^{*}	0.41^{*}	0.31^{*}	0.21^{*}	$0.25^{\ddagger}$
ADAS	-0.44^{*}	-0.42^{*}	-0.34^{*}	-0.22^{*}	-0.18^{**}
MOCA	0.41^{*}	0.35^{*}	0.33^{*}	0.09	$0.29^{\ddagger}$
RAVLT	0.43^{*}	0.34^{*}	$0.27^{\ddagger}$	0.11	$0.20^{\ddagger}$
PsyMEM	0.48^{*}	0.39^{*}	$0.34^{\ddagger}$	0.22^{*}	$0.29^{\ddagger}$
PsyEF	0.45^{*}	0.38^{*}	0.22	0.13^{**}	0.10
HippoVol	0.62^{*}	0.50^{*}	$0.25^{\dagger}$	$0.22^{\dagger}$	0.15^{**}
CDR-SB	-0.33^{*}	-0.35^{*}	-0.35^{*}	-0.30^{*}	-0.31^{*}
DxConv	$--$	17^{*}	17^{*}	$11^{\dagger}$	$10^{\dagger}$
τ	-0.34^{*}	$--$	$--$	$--$	$--$
$p\tau$	-0.35^{*}	$--$	$--$	$--$	$--$
$A\beta42$	0.50^{*}	$--$	$--$	$--$	$--$
$\tau/A\beta42$	-0.46^{*}	$--$	$--$	$--$	$--$
$p\tau/A\beta42$	-0.46^{*}	$--$	$--$	$--$	$--$
APOE	$6.9^{\ddagger}$	$--$	$--$	$--$	$--$
FH	5.01^{**}	$--$	$--$	$--$	$--$

there is poor biomarker specificity of disease progression and when the outcomes contain relatively high amounts of error variance. Determining whether promising treatments are effective in the MCI phase of AD requires accurate identification and inclusion of only those MCI participants most likely to convert to AD and selection of outcomes that are both disease related and possess optimal measurement properties. We have shown that the sample size required to detect a treatment effect can be substantially reduced using the proposed

Table 4.4: CV of rDAm and rDrm vs. MKLm. A: AV45, F: FDG and T: T1GM

Modality	Marker	MCIs	LMCIs	FHMCIs
AV45	MKLm	0.56	0.70	0.42
	rDAm	0.49	0.57	0.41
	rDrm	0.55	0.60	0.46
FDG	MKLm	0.49	0.53	0.39
	rDAm	0.33	0.36	0.26
	rDrm	0.45	0.44	0.30
T1GM	MKLm	0.55	0.60	0.48
	rDAm	0.36	0.42	0.26
	rDrm	0.41	0.42	0.26
A+F	MKLm	0.52	0.63	0.39
	rDAm	0.42	0.49	0.33
	rDrm	0.50	0.57	0.39
A+T	MKLm	0.56	0.67	0.42
	rDAm	0.41	0.49	0.29
	rDrm	0.50	0.51	0.29
F+T	MKLm	0.51	0.58	0.41
	rDAm	0.34	0.38	0.25
	rDrm	0.41	0.44	0.31
A+F+T	MKLm	0.54	0.65	0.39
	rDAm	0.41	0.50	0.28
	rDrm	0.44	0.57	0.30

Table 4.5: Sample sizes with multi-modal baseline rDAm enrichment

Outcome Measure	No enrich	Bottom 20% rDAm$\leqslant$ 0.41	Bottom 25% rDAm$\leqslant$ 0.46	Bottom 33% rDAm$\leqslant$ 0.52	Bottom 50% rDAm$\leqslant$ 0.65
MMSE	1367	200	239	371	566
ADAS	>2000	775	945	>2000	>2000
MOCA	>2000	449	674	960	1919
RAVLT	>2000	591	1211	>2000	>2000
PsyMEM	>2000	420	690	786	1164
PsyEF	>2000	>2000	>2000	>2000	>2000
Hippo Vol	>2000	543	1504	1560	1675
CDR-SB	1586	281	317	430	433
DxConv	895	230	267	352	448

inclusion strategy. The central message of our empirical evaluations is that the multi-modal markers based on our proposed randomized deep network leaning models have good predictive power in identifying future disease progression, as shown in Tables 4.1-4.3 and Figures 4.4-4.5. Together with the rDA's or rDr's capacity to reduce prediction variance (Table 4.4), we see smaller sample estimates compared to existing imaging-derived enrichers, as shown in Table 4.9, across many trial outcomes.

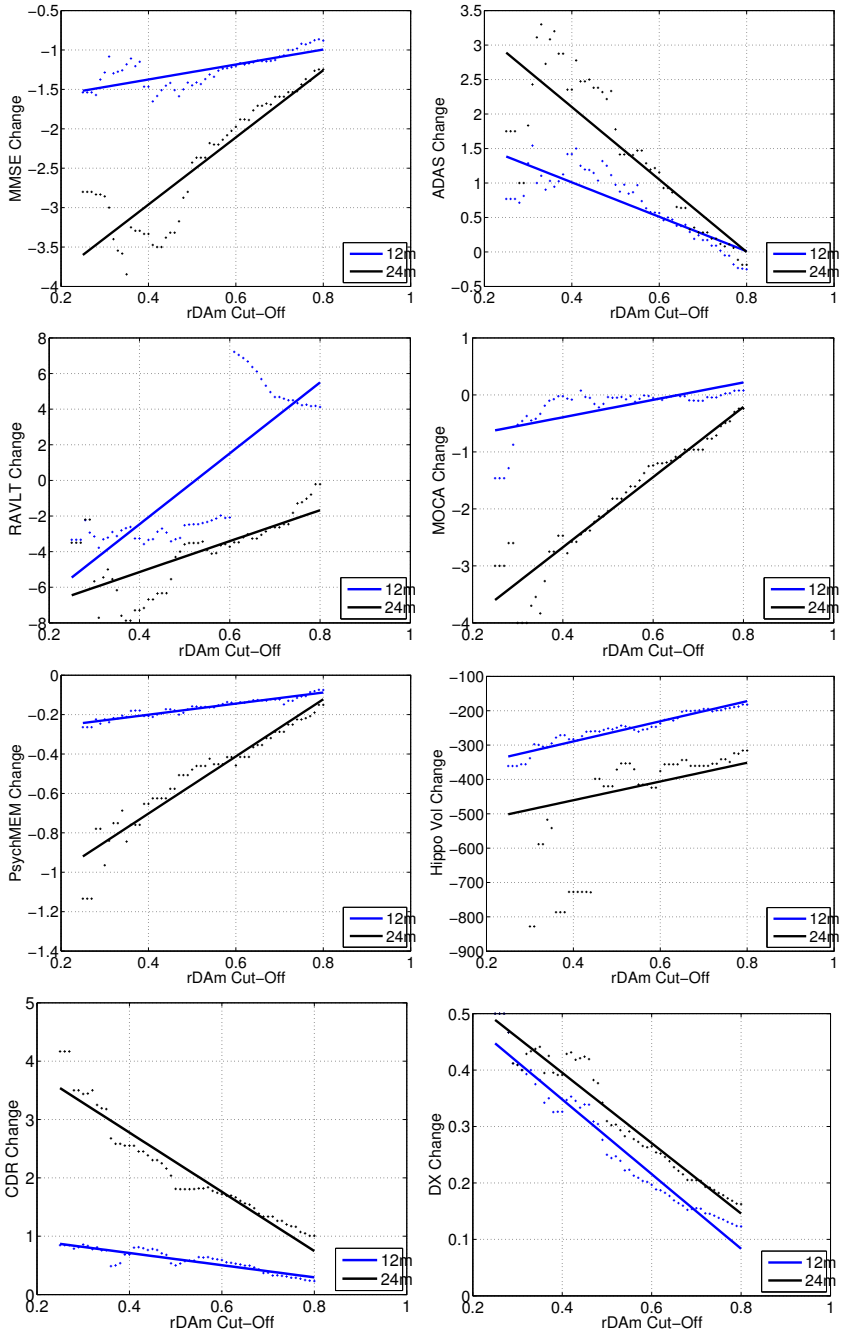


Figure 4.4: Longitudinal Change of measures vs. multi-modal baseline rDAm

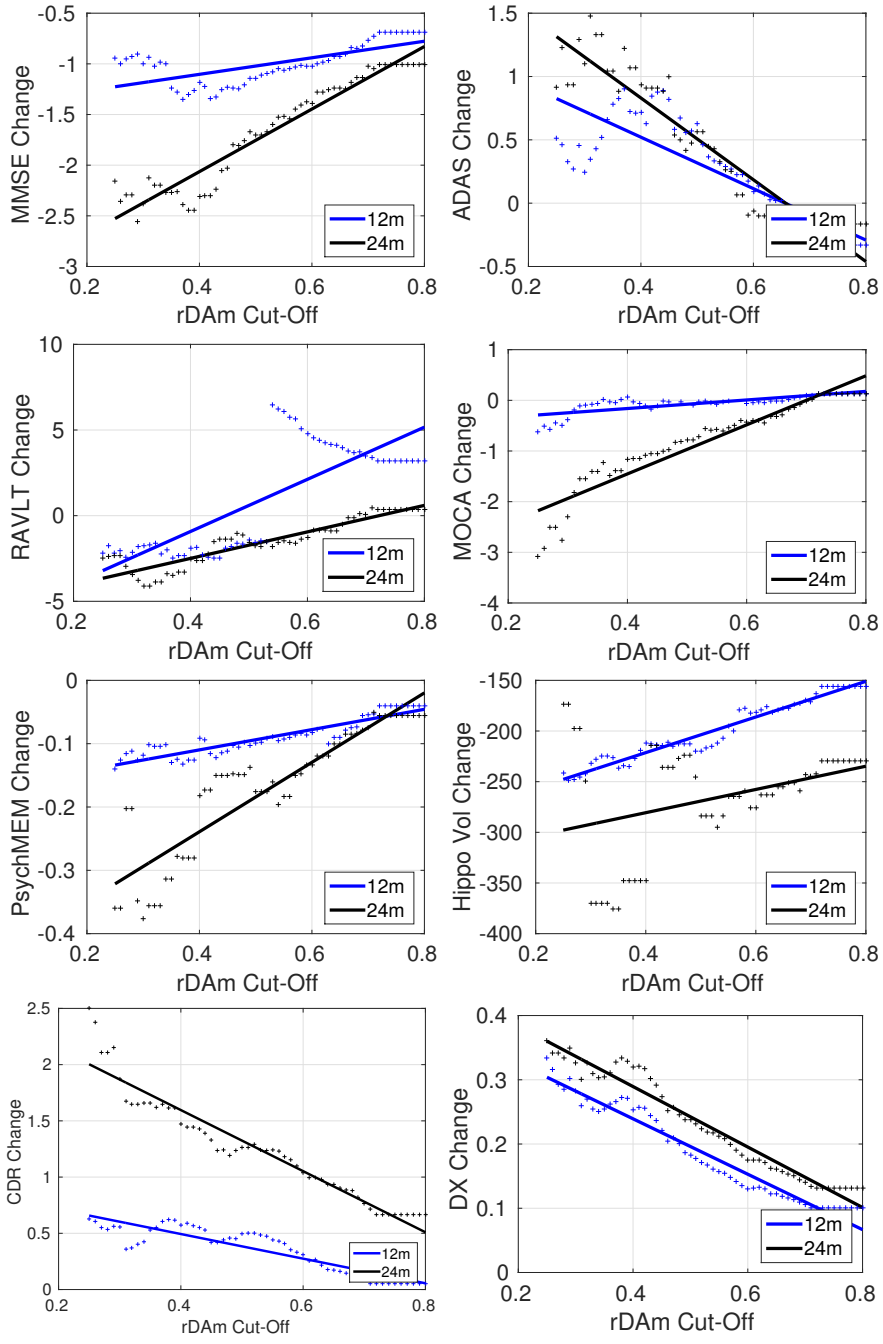


Figure 4.5: Longitudinal Change of measures vs. multi-modal baseline rDrm

Table 4.6: Sample sizes with multi-modal baseline rDrm enrichment

Outcome Measure	No enrich	Bottom 20% rDAm$\leqslant$ 0.39	Bottom 25% rDAm$\leqslant$ 0.44	Bottom 33% rDAm$\leqslant$ 0.58	Bottom 50% rDAm$\leqslant$ 0.70
MMSE	1367	252	341	394	560
ADAS	$>$ 2000	930	1770	$>$ 2000	$>$ 2000
MOCA	$>$ 2000	655	795	1106	1866
RAVLT	$>$ 2000	1556	$>$ 2000	$>$ 2000	$>$ 2000
PsyMEM	$>$ 2000	442	700	799	1215
PsyEF	$>$ 2000	$>$ 2000	$>$ 2000	$>$ 2000	$>$ 2000
Hippo Vol	$>$ 2000	621	1100	1846	$>$ 2000
CDR-SB	1586	307	524	608	618
DxConv	895	232	287	307	429

Table 4.7: Sample sizes with rDAm + FH and/or APOE enrichment

Outcome measure	No enrich	FH Only	APOE Only	rDAm Only	rDAm+ FH	rDAm+ APOE	rDAm + both
MMSE	1367	1668	1015	200	182	240	186
ADAS	$>$ 2000	$>$ 2000	$>$ 2000	775	574	328	271
MOCA	$>$ 2000	$>$ 2000	$>$ 2000	449	516	326	334
RAVLT	$>$ 2000	$>$ 2000	$>$ 2000	591	394	484	332
PsyMEM	$>$ 2000	$>$ 2000	$>$ 2000	420	481	310	333
PsyEF	$>$ 2000	$>$ 2000	$>$ 2000	$>$ 2000	$>$ 2000	1337	721
Hippo Vol	$>$ 2000	$>$ 2000	$>$ 2000	428	391	274	246
CDR-SB	1586	1787	763	281	255	217	225
DxConv	895	932	509	230	244	170	192

Table 4.8: Sample sizes with rDrm + FH and/or APOE enrichment

Outcome measure	No enrich	FH Only	APOE Only	rDrm Only	rDrm+ FH	rDrm+ APOE	rDrm + both
MMSE	1367	1668	1015	252	292	301	306
ADAS	$>$ 2000	$>$ 2000	$>$ 2000	930	1001	1151	642
MOCA	$>$ 2000	$>$ 2000	$>$ 2000	655	669	636	669
RAVLT	$>$ 2000	$>$ 2000	$>$ 2000	1556	1102	$>$ 2000	1544
PsyMEM	$>$ 2000	$>$ 2000	$>$ 2000	442	496	512	385
PsyEF	$>$ 2000	$>$ 2000	$>$ 2000	$>$ 2000	$>$ 2000	$>$ 2000	941
Hippo Vol	$>$ 2000	$>$ 2000	$>$ 2000	621	799	698	698
CDR-SB	1586	1787	763	307	316	351	392
DxConv	895	932	509	232	259	219	239

Tables 4.1-4.3 support the general consensus that imaging data captures disease progression (Hinrichs et al., 2011a; Weiner et al., 2013). This can be seen from the very strong correlations of baseline rDAm and rDrm with cross-sectional and longitudinal changes in several cognitive scores (last four columns). It should be noted that high correlations with hippocampal volume

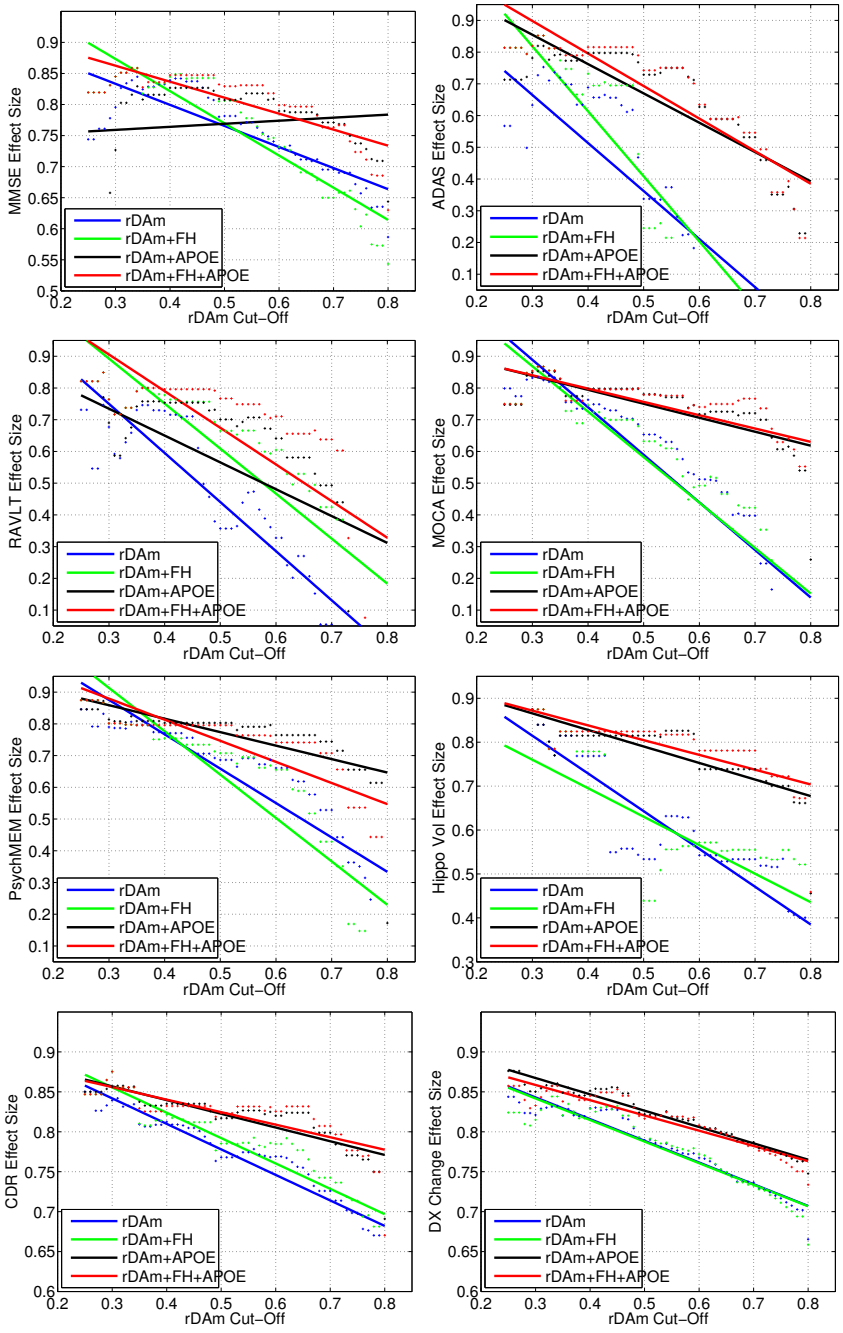


Figure 4.6: Effect Sizes vs. muti-modal baseline rDAm cut-offs

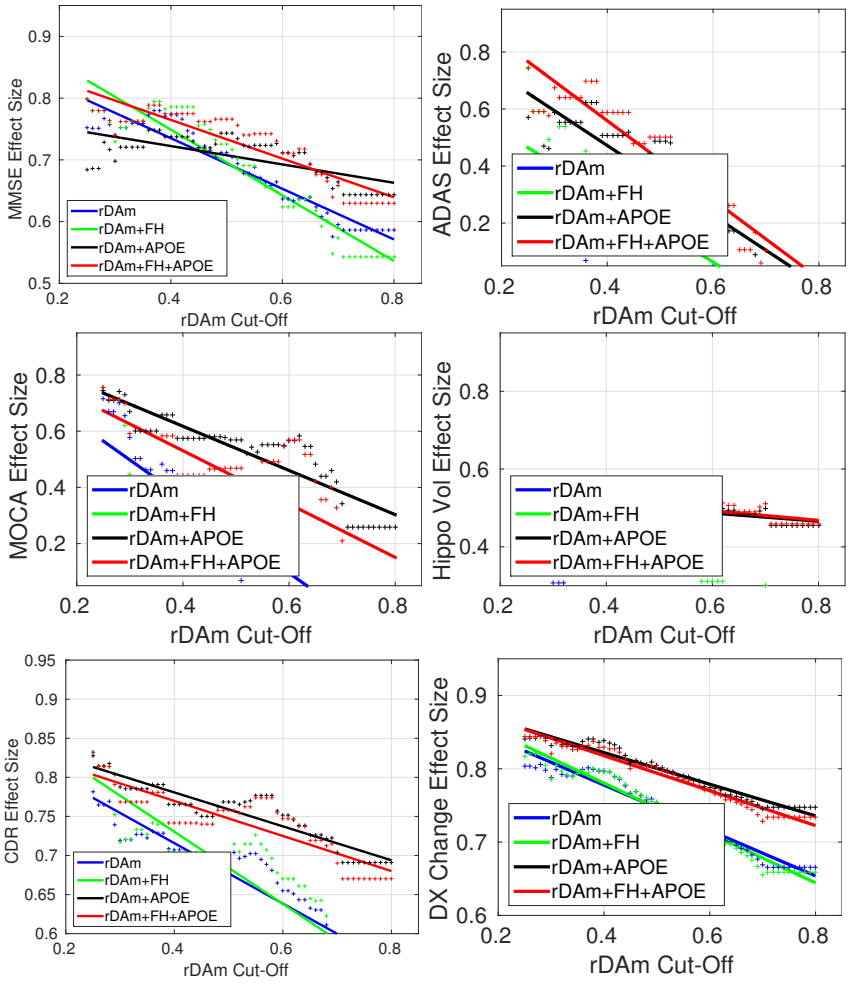


Figure 4.7: Effect Sizes vs. multi-modal baseline rDrm cut-offs

Table 4.9: Multi-modal baseline rDAm and rDrm vs. other enrichers

Sample Enricher	Outcome Measure							
	MMSE	ADAS	MOCA	RAVLT	PsyMEM	HipVol	CDR-SB	Dx
HipVol	540	> 2000	1005	1606	1009	> 2000	389	420
FDG	384	1954	579	> 2000	832	752	415	371
AV45	224	> 2000	875	> 2000	826	698	382	443
FDH	296	> 2000	705	> 2000	826	722	397	402
MKLm	228	874	827	896	487	877	295	284
rDAm	200	775	449	591	420	543	281	230
rDrm	252	930	655	1556	442	621	524	287

across all time-points, which is a structural summary from MRI image, are expected because T1 MRI images at baseline is used in the construction of the markers. Although hippocampus voxels are used in rDA and rDr, its inclusion as an outcome in our experiments is primarily for completeness and continuity with existing AD imaging studies, where it has been extensively studied (Kohannim et al., 2010; Yu et al., 2014; Jack et al., 2013; Weiner et al., 2013). Interestingly, FH had a lower dependence on the baseline markers which might be because its influence is superseded by actual neurodegeneration once a subject reaches MCI stage (i.e., FH may play a much stronger role in the asymptomatic phase). Note that we did not correct for age (and other covariates like brain volume) because the markers reported in Tables 4.1-4.3 are used directly with no covariate correction in our later evaluations on sample enrichment (Tables 4.5-4.8). This is based on the assumption that an actual RCT would not need to correct for the individual's age to evaluate eligibility and the baseline markers are agnostic to all such variables.

Observe that most classification based measures which are used as computational disease markers are generally unbounded (Hinrichs et al., 2012a). These include the prediction score from a SVM based classification model on a test subject (Hinrichs et al., 2011a), or summary measures like S-score, t-score, F-score etc (Kohannim et al., 2010). Unlike these measures, the proposed markers are bounded between 0 and 1, using which we can visualize its predictive power without any post-hoc normalization (as shown in Figures 4.4-4.5). Except for PsyEF, all other measures used as outcomes had steeper changes (in Figure 4.4) over time as baseline rDAm decreased, and in none of the cases was there a clear elbow separating weak and strong decliners. Similar trend is also observed for baseline rDrm. This shows that the disease progression is gradual from healthy to AD, and any classifications (like early and late MCI) are mostly artificial – an observation made earlier in alternate studies on decline (Jack et al., 2013; Weiner et al., 2015; Albert et al., 2011a). Nevertheless, it is of clinical interest to analyze different groups of decliners like late and early MCI, and although the baseline MKLm (the current state-of-the-art in AD classification, (Hinrichs et al., 2011a)) picks up these group differences as well, the proposed models have higher delineation power (Table 4.1). In particular, the p–values for rDAm and rDrm for FH+ vs. FH− case are an order of magnitude smaller than MKLm. These show that, in terms of classification accuracy, rDA and rDr are at least as good as a current state-of-the-art machine learning derived measures.

It is interesting to see rDAm's and rDrm's high predictive power for DxConv (Tables 4.2-4.3 and Figures 4.4-4.5), implying that subjects with smaller baseline rDAm (closer to 0) have very high likelihood of converting from MCI to AD, providing additional evidence that both baseline (and multi-modal) rDAm and rDrm are good predictive disease markers; more so, the predictions from randomized deep networks are good disease markers. Beyond the predictive power, the CVs for proposed models are much smaller than MKLm (Table 4.4) – the central argument that motivated the design of randomized deep networks for sample enrichment (refer Section 4.3). Using CV as a surrogate for prediction variance gives interesting inferences about the stages of the disease, for instance, the CVs for MCIs with FH+ are smaller than that of late MCIs (from Table 4.4). This suggests that a significant number of late MCIs currently have only a mild dementia in terms of both rDAm and rDrm. Most of the prediction power and sample size experiments focused on the multi-modal rDAm and rDrm (using all three modalities – Amyloid and FDG PET and T1 MRI) since several existing studies including (Hinrichs et al., 2011a; Kohannim et al., 2010; Zhang et al., 2011b; Hinrichs et al., 2012b) and many others, and the performance results in Table 4.1, have shown the non-trivial benefit of multi-modal disease markers.

Since the proposed markers are lower bounded to 0 and no elbows are seen in Figures 4.4-4.5, there is no phase change, and we can always select a fixed fraction of subjects that are closest to 0 on the baseline rDAm or rDrm scales, and claim that they are the strong decliners that should be included in a trial. The exact value of such fraction would depend on the logistics and size of the intended trial. This is the reason for the bottom-fraction based enrichment using multi-modal baseline markers as shown in Tables 4.5-4.9. Further, the high predictive power of baseline rDAm and rDrm solves an important bottleneck with existing approaches to designing inclusion criteria that use longitudinal data (e.g., Tensor-based morphometry) (Lorenzi et al., 2010; Hinrichs et al., 2012a). Deploying such methods in practice implies that the trial screening time should be at least a year or longer, which is not practical (both in terms of cost involved and other logistics). Although longitudinal signals are much stronger than cross-sectional ones, the results in Tables 4.1-4.3 and Figures 4.4, 4.5 show that the randomized deep network based markers at trial start-point can still be used with no loss of information, saving trial resources and reducing the cost of trial setup.

A broad observation across Tables 4.5-4.9 is that baseline rDAm is better than rDrm with smaller sample sizes overall, although the trends are the same for

both. Although both SDA and dropout network work with feature denoising, clearly rDA seems to be better at disease prediction as well (Table 4.1-4.3). The higher sample estimates for rDr might be driven by its higher prediction variance (Table 4.4). Few reasons for this broad trend are discussed here. First observe that rDr lacks the unsupervised pretraining step unlike rDA (refer to Section 4.2 and 4.4). Secondly, higher dropout rates might be 'unsuitable' for brain images unlike vision or other machine learning datasets. To see this, observe that although complex interactions across voxels/dimensions are common in brain images, they are nevertheless, registered to a common coordinate space (i.e., unlike object recognition or categorization data, a specific set of voxels in rDA and rDr always correspond to a specific region). In this registered space, the signals on brain images (voxel intensities) are, in general, very weak, and subtle changes in them will correspond to a disease signature (Klöppel et al., 2008; Hinrichs et al., 2011a; Davatzikos et al., 2011; Weiner et al., 2015). For example, the disease signature in hippocampal region corresponds to loss (or dampening) of voxel intensities. Large dropout rates corrupts the images drastically resulting in loss of signal, and in the worst case, the healthy subject might *look very similar to* being a diseased one. Unlike rDr, the post-hoc fine tuning in rDA (which does not involve corruption) compensates for these issues, thereby resulting in better performance. Hence, for the rest of the discussion, we focus mainly on the sample sizes estimated from multi-modal baseline rDAm enrichment instead of rDrm.

MMSE, CDR-SB and DxConv sample estimates (in Table 4.5-4.8) outperform all other alternate outcomes considered here, even in the no-enrichment regime. This may be counter intuitive because of the simplicity of MMSE compared to other composite scores like PsyMEM and PsyEF (neuro-psych memory and executive function composites). It is possible that the composite nature of these measures increases the outcome variance, and thereby increases the sample estimates when used as trial outcomes. Since our population is entirely MCIs, it may be expected that the distribution of baseline rDAms is fairly uniform between 0 and 1, but is not the case as shown from rDAm enrichment cut-offs at each of the four percentiles considered (the top row of last four columns in Table 4.5). More precisely, the bottom 50% corresponds to a cut-off of 0.65 and 33% corresponds to 0.52, which indicates that more than two-thirds of MCIs in the ADNI2 cohort are healthier (i.e., weak decliners), and also that enrichment is important. This idea has also been identified by others using cognitive characteristics (Edmonds et al., 2015). Ideally, we expect to observe a

particular baseline rDAm cut-off (an elbow) at which point there might be the highest decrease in estimates for all outcomes in Table 4.5 and 4.7. The elbow should be a natural threshold point that separates strong and weak decliners on baseline rDAm scale. However, the trends in the last four columns do not seem to suggest such a threshold, which is not surprising following Figure 4.4 and the corresponding discussion above. Specifically, ADAS and RAVLT seem to have an elbow between 25% and 33%, while for MMSE, CDR-SB and DxConv, the elbow is beyond 50%.

Covariate information, or rather, a preliminary selection based on a factor like FH, is almost always helpful in estimating group effects (Table 4.7-4.8). It has been observed that subjects with positive FH (either maternal or paternal) and/or APOE e4 positive may have stronger characteristics of dementia (Huang et al., 2004). This implies that, instead of starting off with all MCIs, it is reasonable to include only those MCIs with positive FH and/or positive APOE e4, and then perform the baseline rDAm or rDrm enrichment on this smaller cohort. APOE had a higher dependence on both rDAm and rDrm compared to FH (from Tables 4.2-4.3), which resulted in smaller sample sizes when using APOE or APOE + FH in tandem with rDAm enrichment (last two columns), than using rDAm + FH (sixth column) for all the cases except MMSE (row 1 in Table 4.7). Note that Table 4.7 corresponds to bottom 20% baseline rDAm enrichment of which about half were FH and/or APOE positive. The strong performance of DxConv with small sample sizes may be because it summarizes the conversion of MCI to AD using longitudinal information, where as rDAm tries to predict this conversion using baseline information alone. We note that, since we have 267 MCIs to begin with, even with rDAm enrichment alone, a bottom 20% enrichment (third column, Tables 4.5 and 4.6) corresponds to a population size of 52, implying that the estimates might be noisy.

Overall, Tables 4.5 and 4.7 support the efficacy of rDAm enrichment (similar trends are seen for rDrm enrichment from Tables 4.6 and 4.8); however, an interesting way to evaluate the strength of rDAm is by fixing the number of trial-enrolled subjects and computing the detectable treatment size (η). If in fact, baseline rDAm successfully selects strong decliners, then the trial should be able to detect smaller expected decrease in disease (i.e., smaller $1-\eta$ or larger η, refer to (4.1)). Figure 4.6 shows exactly this behavior for rDAm (and Figure 4.7 for rDrm), where η (y-axis) increases drastically as rDAm cut-offs (x-axis) are decreased (especially for MMSE, CDR-SB and DxConv). From the perspective of a practitioner, such plots are useful. Specifically, they give tools

for evaluating the minimum treatment effect that can be deemed significant (for the given outcome), from a fixed cut-off and sample size. Such feedback will be helpful to either change the outcomes or change the population size correspondingly.

We discussed in Section 4.1 and 4.3 that although effective imaging-derived disease markers exist (either based on machine learning models or directly computed from imaging ROIs), they may not lead to the best possible clinical trials. This is supported by the results in Table 4.9, where both rDAm and rDrm (designed to explicitly reduce the prediction variance) are compared to existing markers that have been used as trial inclusion criteria (Grill and Monsell, 2014; Kohannim et al., 2010; Yu et al., 2014). For example, ROI summaries from multiple imaging modalities have often been used as trial enrichers (Grill et al., 2013; Grill and Monsell, 2014), and rDAm and rDrm significantly outperform these baselines (first four rows in Table 4.9). Further, (Kohannim et al., 2010) used SVMs to design an effective disease marker and used it as an inclusion criterion in trials. Correspondingly, we compared rDAm and rDrm to MKLm (based on a multi-kernel SVM), and the results in Table 4.9 show that baseline rDAm and rDrm as enrichers outperform MKLm, and the improvements are higher for MOCA, RAVLT and Hippocampal Volume as outcomes. For our experiments, we did not adjust any of the parameters relative to the results reports earlier (Hinrichs et al., 2011a), and they were the defaults for the MKL code-base provided on the webpage (`http://pages.cs.wisc.edu/~hinrichs/MKL_ADNI/`) by the authors.

The necessity of incorporating multi-modal information in designing any disease markers has been reported earlier (Hinrichs et al., 2011a; Zhang et al., 2011b). This is further supported by the improvement of rDAm estimates over uni-modal measures including hippocampal volume, FDG ROI summaries and florbetapir ROI summaries. These results also build upon the work of (Grill et al., 2013; Grill and Monsell, 2014) where such unimodal imaging summaries are used for enrichment. It is possible to demonstrate that the performance gains of rDAm over (Grill et al., 2013; Grill and Monsell, 2014) is not merely due to using three distinct modalities but also heavily influenced by the underlying machine learning architecture that exploits this information meaningfully. To see this, compare our proposed markers to the enricher "FAH" which combines three uni-modal measures, FDG, florbetapir, and hippocampal volume in Table 4.9. FAH's sample estimates are still larger than those obtained from both rDAm and rDrm, implying that the reductions are not merely due to multi-modal

data or small population size, but due to the efficacy of the method introduced here, and their capacity of picking up strong decliners with high confidence and small variance.

Overall, our evaluations and the resulting trends clearly suggest that rDAm and rDrm enrichment (rDAm better among them) reduce sample sizes significantly leading to practical and cost-effective AD clinical trials. The rDA and rDr models scale to large dimensions, learn from only a small number of instances, and can be easily incorporated to design robust multi-modal imaging (or non-imaging, if the corresponding blocks are designed appropriately) markers. The full implementation of the framework is made available at `http://pages.cs.wisc.edu/~vamsi/rda`. The framework can nevertheless be improved further, particularly in terms of using richer pooling strategies instead of simple ridge regression, using covariate information like age, CSF levels (or FH, APOE etc.) in the network construction itself (instead of the pre-selection setup used earlier in our experiments). An interesting extension would be to incorporate multi-modal and multi-domain (e.g., ordinal, continuous and nominal) information directly into the rDA or rDr construction leading to multi-variate randomized deep network models.

Chapter 5

Enricher Design for Multi-site Trials

5.1 Introduction

In the previous chapter we showed that neural networks are suitable for designing computational enrichers in clinical trials. In this chapter, we study the setup where the clinical trial comprises of multiple centers/sites. This is a common aspect of large-scale international clinical trials. Observe that unlike support vector machines or random forests, the critical component of deep networks is the choice of the architecture. There are a lot of empirical validation strategies, for selecting a good deep network for a given problem, however for practical deployability, and usage by non-experts, there are very few guidelines available for choosing the structure. Most importantly, for applying the ideas presented in Chapter 4, we need more theoretically driven algorithmic strategies for choosing the relevant neural network for the given task. We will motivate this contribution from an alternate point of view, and develop the presentation to be as general as possible with application beyond the specific topic of this thesis. The question we ask is – *how to design a deep architecture for a given problem/dataset*. Specifically, given the dataset that we want to learn from, what would be some useful network structures that a non-expert can choose to start working with. The proposed framework here is applicable for learning tasks on any dataset beyond medical studies (like ADNI and WRAP studies); and the proposed analysis procedure is more theoretical and will be easier to present agnostic of any downstream application. After introducing the problem and motivation, we will relate back the solution procedure and the resulting proposal to the original problem of multi-center medical studies.

The ingredients for designing deep networks

The successful deployment of deep learning algorithms in a broad spectrum of applications including localizing objects in images (Lenz et al., 2015; Krizhevsky et al., 2012; Szegedy et al., 2014), analyzing particle accelerator data (Baldi et al., 2014), converting speech into text (Hinton et al., 2012), learning to play video games (Mnih et al., 2013), predicting the activity of drug molecules (Lusci et al., 2013; Sutskever et al., 2014) and designing clinical trials (Plis et al., 2014; Ithapu et al., 2015b) provides compelling evidence that they can learn complex concepts with fairly minimal feature engineering or preprocessing. This success is attributed to the idea of composing simple but non-linear modules that each transform the lower levels (i.e., raw or normalized data) into a representation at a more abstract level (LeCun et al., 2015; Bengio, 2009a; Bengio et al., 2013a). These layers of features are not hand designed, rather learned from data using a general learning process guided by the structure of the carefully chosen non-linear modules. While this high level representation learning procedure is quite general, the problem at hand may at least partly govern the choice of the architecture and require certain modifications to the algorithm. Consequently, motivated by various experimental considerations one encounters in practice, several variants of deep architectures and corresponding regularization schemes have been developed over the past decade (Lee et al., 2009; Vincent et al., 2010; Baldi and Sadowski, 2014; Goh et al., 2013; Wan et al., 2013; Ioffe and Szegedy, 2015).

Complementary to such design, algorithmic, and empirical developments, there is also a growing recent interest in better understanding the mathematical properties of these models. Over the last few years, several interesting results have been presented (Castillo et al., 2006; Shao and Zheng, 2011; Dauphin et al., 2014; Livni et al., 2014; Bach, 2014; Arora et al., 2014; Patel et al., 2015; Janzamin et al., 2015; Hardt et al., 2015). While some of these studies address the hypothesis space, and the corresponding sets of functions learnable by deep networks, others analyze the nature of the parameter space that needs to be learned via backpropagation. A more detailed discussion about these works is included in Section 5.1 where we review some of the most related papers. This literature is still rapidly evolving, and there are a large number of interesting, and open, questions listed, for instance, in (Bengio, 2012; Wang and Raj, 2015) whose answers will help guide, going forward, the training, debugging and designing of large-scale, and often very deep, multi-layer neural networks for

arbitrary and complex learning tasks. A few such motivating questions are given below that will help provide the context for this work.

(Q1) There is strong empirical evidence that changing the network depth or layer lengths (i.e., sizes of the individual layers) leads to significant and, at times, a non-trivial change in generalization performance (measured via testing set error) (Bengio, 2009a; Hinton and Salakhutdinov, 2006; Simonyan and Zisserman, 2014; Szegedy et al., 2014). To that end, is it possible say anything specific about the *best* possible architecture for learning representations from a given dataset or for a specific task? In other words,

- **(a)** Can we explicitly compute the influence of the number of layers, the corresponding lengths and activation function choices, on the complexity of the input-to-output mapping being learned?
- **(b)** Or, how do the types of learnable mappings vary as we change the lower and/or higher layer lengths? Are there sets of appropriate and 'good' architectures for a given learning task?

(Q2) Does fully supervised dropout (Srivastava et al., 2014) compensate for unsupervised pretraining (Erhan et al., 2009)? Are these two ideas theoretically related? As a function of the network structure, are there regimes where one is intrinsically better?

(Q3) Which networks have faster parameter convergence? A recent result on stochastic gradients relates the training time to generalization (Hardt et al., 2015). On new learning tasks (with no established benchmarks), can the notion of parameter convergence, in tandem with (or without) such a generalization result, be used as a criteria for choosing the best architecture from a family of networks?

(Q4) Are there concepts that *cannot* be learned by a given deep model? Specifically, given the form of non-linearities used in a design, can we say much about the type of data statistics that the network is guaranteed to learn?

(Q5) Given that the choice of non-linearities is mainly task dependent, for instance, max-pooling guarantees some form of translation invariance (Nagi et al., 2011). In general, these choices are not based on the eventual goal of convergence or generalization – and instead such non-linearities can be interpreted as efficient regularizers. So,

- **(a)** Given the task, are there optimal choices for such non-linearities? More importantly, using task-specific information and the data distribution, are there strategies to *choose* the best activation from a given

bag of non-linearities?

(b) Are there strategies for designing activations beyond convolutions and translation invariance, for instance, to efficiently learn models for applications where the data are brain images (e.g., magnetic resonance images (Frisoni et al., 2010; Killiany et al., 2000)) and genetic data (e.g., single nucleotide polymorphisms (Sachidanandam et al., 2001)), that are presumed to lie on complex combinations of low-dimensional manifolds (Gerber et al., 2009; Corouge et al., 2004; Lazar et al., 2003).

(Q6) Different families of deep networks have been shown to lead to similar empirical performance (Livni et al., 2014; Ngiam et al., 2011). When can we say that two arbitrary deep networks are *equivalent* to each other in terms of the hypothesis spaces they can model? Can one construct transformations or maps between such families?

(Q7) One of the reasons behind the success of deep networks is the availability of large amounts of data (LeCun et al., 2015; Bengio et al., 2013a; Erhan et al., 2009; Livni et al., 2014). But can we also successfully perform small sample size deep learning with dataset sizes much larger than the number of features (common in biomedical applications involving expensive acquisitions)? Or, can the network be *regularized* to compensate for the lack of sufficient amount of training data?

For some of these problems, there is good empirical evidence but little theoretical support (e.g., **Q1** and **Q6**), while some of the other questions, to the best of our knowledge, are yet to be carefully studied and addressed from the theoretical perspective. For instance, while this literature is rapidly evolving, not much is known about the relationship of the network architecture and the input distributions (**Q1**, **Q3** and **Q5**), in the context of algorithms for small sample size deep learning (**Q7**).

The broad goal of this work is to address some of these open problems, or at the least, provide a good starting point towards answering them. We pursue this goal by presenting a potentially useful theoretical characterization of certain convergence properties of deep networks, from the perspective of parameter estimation. At the high level, we study the relationship between learnability i.e., convergence of parameter estimation in an optimization sense, and the architecture of the deep networks for a given training dataset. In particular, we are interested in the interplay of the *network architecture* and *data statistics* and their influence on the *learning schemes*. These aspects of deep networks, although intuitively seem related, have, for the most part, been

studied separately. Specifically, (Dauphin et al., 2014; Livni et al., 2014; Bach, 2014) and others address the optimization of deep networks, while (Lee et al., 2009; Baldi and Sadowski, 2014; Goh et al., 2013; Wan et al., 2013) and others deal with the regularization aspects – an *explicit* characterization of their interaction is lacking in the literature so far. We describe how characterizing this *interplay* directly enables answering several of the above questions, which may facilitate or guide other empirical investigations.

Overview: The most commonly used procedure for parameter estimation in deep networks is the mini-batch stochastic gradients (Bottou, 1991; LeCun et al., 2012; Bengio, 2012). This involves deriving the average gradient from the error computed on the learning objective (or loss) using a few training instances, and adjusting the weights/parameters accordingly. This continues for a fixed number of iterations and one can, in principle, perform some checks at the stopping iteration and repeat the procedure, if needed. Our general goal is to tie the mechanics of this procedure, to the network structure. In contrast to other recent results derived independently of our work (Janzamin et al., 2015), we directly work with stochastic gradients with no strong assumptions on the data/network structure (Bach, 2014), which enables obtaining results that are quite generally applicable. The starting point of our analysis is a recent work by (Ghadimi and Lan, 2013) dealing with the convergence of stochastic gradients for arbitrary nonconvex problems using a first-order oracle. We build upon and adapt this analysis by first addressing single-layer networks and unsupervised pretraining, and then, the more general case of multi-layer dropout networks, followed by convolutional and recurrent neural networks. In each of these cases, once the network structure is tied to the behaviour of the gradients, we analyze the influence of input data statistics on the parameter estimation and convergence. More importantly, apart from addressing the interplay, the algorithms we present, with minor tweaks, are easily deployable to the standard training pipeline. The bounds natively take into account the standard regularization schemes like dropout and layer-wise pretraining, making them even more useful in practice.

The design choice problem

Within the last several years, several variants of network architectures have been proposed with various combinations of fully connected (e.g. Boltzmann machines (Nair and Hinton, 2010) or autoencoders (Vincent et al., 2010)), convo-

lutional or recurrent layers with non-linearities like rectified linear units (Dahl et al., 2013), maxout (Goodfellow et al., 2013) and/or max-pooling (Nagi et al., 2011) trained using dropout (Srivastava et al., 2014), dropconnect (Wan et al., 2013). The goal of a practitioner is to choose the network architecture most suitable for solving a given problem/application. One is then faced with a 'multitude' of architectural choices when trying to decide the specific construction that is likely to work the best. For instance, one may decide to use a network with combinations of convolutional and fully connected layers using rectified linear units where learning is performed using dropout, or prefer other variations of such a *prototypical* construction. The choice is, at least, in part, governed by the domain knowledge apart from other resources constraints including the sizes of available datasets and/or the desired convergence (or generalization) of the estimated parameters. Therefore, this interplay of (network) structure and parameter convergence is important to guide the choice of which setup will be most useful. Consider the following simple issues that comes up routinely in practice.

1. *From the prototypical network to the desired one:* With the goal of achieving the best possible generalization via validation set error or some other performance measure of interest, one may start with a prototypical network described in the literature and then *modulate* it by changing the depth, layer lengths, or the learning algorithm (Wan et al., 2013), (Goodfellow et al., 2013). The specific modulations, i.e., the design choices, are mostly driven by domain knowledge or user expertise. For a wide variety of concepts/tasks, which non-linearities to choose may not be clear, and may have to be based on trial/error.
2. *How many designs to check for?* Beyond the structural adjustments that might be necessary, there is a separate question focusing on how many adjustments to evaluate for and when to stop. For smaller and medium sized datasets, a few dozen, if not a hundred, such modulated models may be tested. For much larger datasets, in general, the biggest model that can reasonably fit in the memory, and can be trained, is selected. There are not many guidelines available to facilitate this process.

Clearly, any best practices, e.g., basic principles or "ground rules", in order to select the *appropriate* families of networks and the corresponding structural modulations for customization, will be beneficial. A systematic or informed (based on some set of rules) network design strategy can address some of these problems, while also providing insights into design choice issues beyond the

type and number of models to search for.

1. *Resource allocation:* Given a learning task that is relevant in an application, criteria for the dataset size that one must collect within a study is very useful in biomedical applications where the acquisition is costly and time-consuming — the budget must be explicitly justified to funding agencies. For instance, a single acquisition of positron emission tomography (PET) scan can cost $\$3000+$, while a MRI scan may cost $\sim \$500$. Similar to power calculations in statistical analysis, a sensible strategy for estimating the dataset size required to achieve a certain level of parameter convergence is highly desirable. Further, training 'larger than necessary' networks will entail more computing and financial resources. A simple way to quantify such gains is by asking the question when to stop the stochastic gradient update procedure? Cutting down the minimum number of such training iterations required to achieve a certain level of convergence or generalization, even by $5 - 10\%$, will result in quantifiable computing and financial savings, for instance, when learning networks on cloud platforms.
2. *Model family/class selection:* It seems that searching for the most appropriate model will benefit from having a mechanism that lists out the families of networks with similar (expected) generalization for the given task. For instance, fully connected sigmoidal networks might be more appropriate than convolutional layers for certain tasks where the input feature space is registered (or normalized) to some "template" space. Such co-registered data is common in speech modeling (Hinton et al., 2012), medical imaging (Hinrichs et al., 2011b) etc.
3. *Hyper-parameter selection:* The learning hyper-parameters are generally selected via cross validation or some additional domain knowledge/expertise. Although, automated hyper-parameter tuning has been studied using ideas from Bayesian Optimization (Snoek et al., 2012) and Bandit learning (Li et al., 2016a), there is little guidance available for adapting such procedures for deep learning. It is not yet clear if the design choice problem will benefit from such strategies — restricting the hyper-parameter space using information/constraints on structure and the achievable convergence is one potentially useful way to address this issue.

In this chapter, we discuss how the systematic design strategies may fall out of a framework that analyzes the interplay of structure and parameter

convergence. While we do not claim that the network parameters should be decided solely based on the convergence; instead, we show that convergence will enable *assessing* the goodness of a chosen network, and therefore, in turn, the structure or learning hyper-parameters can be changed as necessary. On a new dataset/task, one may first 'select' the best possible network structure guided by the results described in this paper, using information from data statistics and other domain knowledge. The trends will help adapt or modulate the chosen network as needed, based on certain downstream performance measures like accuracy on validation sets.

Analysis of Data from Multi-center Studies

The design choice problem is more involved where models need to be constructed on datasets collected from multiple sources or acquisition sites. This problem can often be tackled *prospectively*, e.g., within large scientific studies (Klunk et al., 2015) across multiple sites that are becoming more prevalent to better understand disease progression (Mueller et al., 2005) or evaluating the efficacy of drugs (Sperling et al., 2014). Data sharing/pooling considerations are usually formalized within the project guidelines, which makes the follow-up analysis tasks relatively convenient. In contrast, in many cases, the decision to *pool* data sets across multiple sources or centers is *retrospective* — in an effort to increase the sample size of certain statistical tests (Zhou et al., 2016), so that specific scientifically interesting hypotheses can be evaluated (e.g., the data at each site, by itself, may be under-powered due to smaller sample size). Separate from statistical analysis, we may seek to train richer machine learning models for prediction or other clinical purposes on such pooled datasets. However, because of privacy laws that differ between countries (e.g., Europe, United States), it may not always be possible (or may be logistically difficult) to pool/transfer data collected at different sites. One may perform meta-analysis (Haidich, 2011) to aggregate individual models inferred from different datasets/sites. Alternatively, one can choose to run or train 'similar' statistical models at each site independently (respecting each country's data sharing laws), and later combine the models or the estimates (e.g., weighted by the sample sizes or the moments of the sample distribution).

Assuming that the models of interest are rich classes of neural networks, the ability to learn the *same or comparable* networks across all the datasets is very useful — it makes the downstream aggregation or interpretation task

significantly simpler. If the underlying physical characteristic of the datasets are consistent, it makes sense to learn comparable deep network models across multiple (similar) datasets. This will enable models that are *transferable* across multiple data centers ensuring that similar network models, with similar degrees of freedom, will be constructed on both data acquisition sites, respecting geographical data transfer constraints. Observe that an inherent component of statistical estimation is to assess the confidence of the produced estimates, typically, via computing the error bars (or confidence intervals) using some bootstrapping with/without replacement. Clearly, for this bootstrapping to be sensible, each realization, i.e., each set of deep networks from all the sites, will need to be learned up to the "same" level of convergence or generalization. In this way, any posthoc statistical procedure, like field of experts (Roth and Black, 2005), non-nested model selection (Pesaran and Weeks, 2001), or simply the information criteria like AIC, BIC etc. (Arlot et al., 2010), can then be used to select the best or aggregated model across the datasets/sites. Our framework will facilitate the multi-center design choice problem, and also provide guidance on how the bootstrapping will be performed in practice. The ideas proposed in this chapter are summarized in Ithapu et al. (2016) and Ithapu et al. (2017b).

We point out a caveat here. The questions we are posing here are difficult and is unreasonable to expect a complete solution within the scope of this work (and this thesis). Instead our **main contributions** listed here will take a step towards addressing the above listed questions. **(a)** Motivated by the recent idea of randomly stopping gradient updates from (Ghadimi and Lan, 2013), we present a framework for analyzing mini-batch stochastic gradients on arbitrary multi-layer deep networks. We prove gradient convergence of multi-layer networks learned via dropout with/without layer-wise pretraining. **(b)** Building upon the framework, we derive explicit relationships between the network structure, gradient learning parameters and input data statistics. Our results are consistent with many empirical studies, but further *guide* the choices of network/learning hyper-parameters for modeling a given data set. **(c)** We present extensive experiments evaluating and visualizing the trends from the derived bounds. **(d)** We present systematic design procedures for constructing deep networks that achieve a certain level of convergence and generalization, and discuss a case study using such optimal design choices in learning deep networks on medical imaging data.

Related Work

The body of literature addressing backpropagation in neural networks, and deep networks in general, is vast and dates back at least to the early 1970s. Here, we restrict the discussion only to those works that fall immediately within the context of this paper. While there are a number of early seminal papers addressing variants of backpropagation and stochastic gradients, and studying the convergence of neural networks training (Becker and Le Cun, 1988; LeCun et al., 1998; Vogl et al., 1988; Magoulas et al., 1999), a number of recent works (Ngiam et al., 2011; LeCun et al., 2012; Bengio, 2012; Dauphin et al., 2014; Janzamin et al., 2015; Hardt et al., 2015; Andrychowicz et al., 2016) provide a fresh treatment of these problems and analyze efficient learning schemes in the context of deep networks specifically. The solution space of backpropagation in this setting has also been addressed in recent results (Castillo et al., 2006; Shao and Zheng, 2011), providing new insights into the types of functions learnable by deep networks (Livni et al., 2014; Bach, 2014). (Hardt et al., 2015) have shown that better generalization can be achieved by ensuring smaller training times, while (Dauphin et al., 2014) describe, in detail, the non-convex landscape of deep learning objectives and the goodness of local optima. Beyond these optimization related works, several authors have independently addressed the regularization and learnability aspects of deep networks. For example, (Wager et al., 2013; Baldi and Sadowski, 2014) extensively analyzed the properties of dropout learning, and (Patel et al., 2015) develops a comprehensive probabilistic theory of deep learning. (Arora et al., 2014) study the existence and construction of deep representations by exploiting the structure of the network, and (Arora et al., 2015) presents a generative model for ReLU type deep networks with the assumption that the network weights are random. The number of linear regions computable by deep networks is studied by (Montufar et al., 2014). Very recently, (Wei et al., 2016) have studied the equivalence of arbitrary deep networks. Complimentary to these, (Janzamin et al., 2015) uses a tensor decomposition perspective to offer guarantees on training certain types of networks.

Hyperband Algorithm: The problem of choosing the appropriate model for the given problem relates to the broader set of algorithms targeted towards hyperparameter search. Most such search methods are Bayesian by nature (Feurer et al., 2015; Snoek et al., 2012). The authors of (Li et al., 2016b) proposed a more efficient hyperparameters search based on adaptive resource allocation inspired by concept of bandit learning (Katehakis and Veinott Jr, 1987). Hy-

perband relies on a principled early-stopping strategy to allocate resources, allowing it to evaluate orders of magnitude more configurations than black-box procedures like Bayesian optimization. As mentioned earlier, the proposed framework will involve developing a new analysis framework and computing bounds that relate several hyperparameters corresponding to the structure and learning of the network. These bounds in turn lead to design criteria – and so we use the Hyperband algorithm as a baseline while presenting our framework. Observe that hyperband is a general hyperparameter search procedure, while the goals of our bounds is to specifically target convergence and structural aspects of deep learning. We will also discuss these issues more in Section 7.1.

5.2 Preliminaries

Our goal is to relate the structural aspects of arbitrary deep networks (including the number of layers, the length of each layer and activation function types) to the corresponding learning hyper-parameters (like dropout rates, gradient iterations, stepsizes, and so on) using non-trivial information about the input data distribution (e.g., moments). We first present some basic notations to setup the analysis and then provide a roadmap to our framework before describing it in the next sections. The proofs for all the technical results are provided in Section 5.9.

Notation. Let $\mathbf{x} \in \mathbb{R}^{d_x}$ and $\mathbf{y} \in \mathbb{R}^{d_y}$ denote the input feature vector and the corresponding output (or label) respectively. Given multiple $\{\mathbf{x}, \mathbf{y}\} \in \mathcal{X}$, the unknown input-to-output mapping is modeled by a L-layered neural network (L-NN). An L-NN comprises the input (or visible) unit $\mathbf{x}$, followed by $L-1$ hidden representations $\mathbf{h}^1, \ldots, \mathbf{h}^{L-1}$ and the output (or final) unit $\mathbf{y}$ (Bengio, 2009a). The lengths of these $L+1$ representations are $d_0 = d_x, d_1, \ldots, d_{L-1}, d_L = d_y$ respectively. Each layer transforms the representations from the previous layer by first applying an affine transform, followed by a non-linear function which may be non-convex (in general) but not necessarily point-wise (Bengio, 2009a; Bengio et al., 2013a). The layer-wise transformation matrices are denoted by $\mathbf{W}^l \in \mathbb{R}^{d_l \times d_{l-1}}$ for $l = 1, \ldots, L$. The hidden representations are given by $\mathbf{h}^l = \sigma(\mathbf{W}^l, \mathbf{h}^{l-1})$ for $l = 1, \ldots, L-1$ ($\mathbf{h}^0 = \mathbf{x}$), and the output layer is $\mathbf{y} = \sigma(\mathbf{W}^l, \mathbf{h}^{L-1})$, where $\sigma(\cdot)$ represents the non-linear function/mapping between layers. For a single-layer network with no hidden layers, we have $\mathbf{y} = \sigma(\mathbf{W}, \mathbf{x})$ where $\mathbf{W}$'s are the unknowns. Note that the bias in the affine transformation is

handled by augmenting features with 1 whenever necessary. The distributional hyper-parameters of interest are $\mu_{\mathbf{x}} = \frac{1}{d_x}\sum_j \mathbb{E}x_j$ and $\tau_{\mathbf{x}} = \frac{1}{d_x}\sum_j \mathbb{E}^2 x_j$, which correspond to the average first moment and average squared first moment of the inputs respectively (the average is across the d_x dimensions). For simplicity we assume $\mathbf{x} \in [0,1]^{d_x}$ and $\mathbf{y} \in [0,1]^{d_y}$, and so $\mu_{\mathbf{x}} \in [0,1]$ and $\tau_{\mathbf{x}} \in [0,1]$.

Consider the following minimization performed via mini-batch stochastic gradients (Bottou, 2010),

$$\min_{\mathbf{W}} \quad f(\mathbf{W}) := \mathbb{E}_{\mathbf{x},\mathbf{y}}\mathcal{L}(\mathbf{x},\mathbf{y};\mathbf{W}) \tag{5.1}$$

where $\mathcal{L}(\cdot)$ denotes some loss function parameterized by $\mathbf{W}$ and applied to data instances $\{\mathbf{x},\mathbf{y}\}$. Denote $\eta := \{\mathbf{x},\mathbf{y}\} \sim \mathcal{X}$. The mini-batch stochastic gradient update using B samples $\eta^1,\ldots,\eta^B$ and gradient stepsize γ is

$$\mathbf{W} \leftarrow \mathbf{W} - \gamma G(\eta;\mathbf{W}) \tag{5.2}$$

where the gradient $G(\eta;\mathbf{W})$ computed at $\mathbf{W}$ using the sample set $\eta^1,\ldots,\eta^B$ is given by

$$G(\eta;\mathbf{W}) = \frac{1}{B}\sum_{i=1}^{B} \nabla_{\mathbf{W}}\mathcal{L}(\eta^i;\mathbf{W}) \tag{5.3}$$

Depending on $\mathcal{L}(\cdot)$, the expression in (5.1) corresponds to backpropagation learning of different classes of neural networks using stochastic gradients. To address many such broad families, we develop our analysis for three interesting and general classes of deep networks, starting with single-layer networks, followed by unsupervised pretraining via box-constrained denoising autoencoders (Vincent et al., 2010), and finally multi-layer deep networks with dropout (Srivastava et al., 2014), both the layer-wise pretrained and fully supervised versions. For each of these settings, the loss function $\mathcal{L}(\cdot)$ is defined below and additional details for these classes of networks can be found in (Bengio, 2009a; Vincent et al., 2010; Srivastava et al., 2014).

1**-NN, Single-layer Network**:

$$\mathcal{L}(\mathbf{x},\mathbf{y};\mathbf{W}) = \|\mathbf{y} - \sigma(\mathbf{W}\mathbf{x})\|^2 \tag{5.4}$$

where $\mathbf{W} \in \mathbb{R}^{d_y \times d_x}$. One can induce feature denoising into this 1-NN learning via a dropout scheme (Srivastava et al., 2014). We discuss this single-layer dropout network shortly after presenting the multi-layer loss functions.

DA, Box-constrained Denoising Autoencoder:

$$\mathcal{L}(\mathbf{x}, \mathbf{y}; \mathbf{W}) = \|\mathbf{x} - \sigma(\mathbf{W}^T \sigma(\mathbf{W}(\mathbf{x} \circ \mathbf{z})))\|^2 \quad ; \quad \mathbf{W} \in [-w_m, w_m]^{d_h \times d_v} \tag{5.5}$$

where $\circ$ denotes element-wise product and $\mathbf{z}$ is a binary vector of length d_x. Given the denoising rate $\tilde{\zeta}$, the binary scalars $z_1, \ldots, z_{d_x} \sim \text{Bernoulli}(\tilde{\zeta})$ are indicators for "nullifying" the input layer units, i.e., whenever $z_i = 0$, the i^{th} element of the input in forced to be equal to 0. Denoting $\mathbf{x} \circ \mathbf{z}$ as $\tilde{\mathbf{x}}$, this implies that $\tilde{\mathbf{x}}_i = 0$ whenever $z_i = 0$. Here, $[-w_m, w_m]$ is the box-constraint on the unknowns $\mathbf{W}$, which forces the learned representations to *not* saturate around 0 and/or 1, and has been widely used in variants of both autoencoder design and backpropagation itself. The nature of this constraint is similar to other regularization schemes like (Bengio, 2012) and (Ngiam et al., 2011; Srivastava et al., 2014). Although feature dropout based fully-supervised models have been shown to achieve good generalization accuracy, unsupervised *pretraining* was important in many early papers (Bengio, 2009a; Erhan et al., 2010b). Here, we analyze DA not only due to its broad usage for pretraining (which was popular a few years back), but also because of its relationship to dropout — both schemes inject noise into the input and/or hidden features.

L-NN Multi-layer Network:

$$\begin{aligned} \mathbf{h}^0 = \mathbf{x}; \quad \mathbf{h}^l &= \sigma\left(\mathbf{W}^l(\mathbf{h}^{l-1} \circ \mathbf{z}^l)\right) \\ \mathcal{L}(\mathbf{x}, \mathbf{y}; \mathbf{W}) &= \|\mathbf{y} - \sigma\left(\mathbf{W}^L(\mathbf{h}^{L-1} \circ \mathbf{z}^L)\right)\|^2 \end{aligned} \tag{5.6}$$

where $l = 1, \ldots, L-1$. Recall the dropout scheme in training deep networks which focuses on the overfitting problem (Srivastava et al., 2014). In each gradient update iteration, a random fraction of the hidden and/or input units are dropped based on the (given) dropout rates $\zeta_0, \ldots, \zeta_{L-1}$ (Srivastava et al., 2014). Similar to DA, the dropped out units for each layer are denoted by a set of binary vectors $\mathbf{z}^1, \ldots, \mathbf{z}^L$ such that $\mathbf{z}^l_i \sim \text{Bernoulli}(\zeta_l)$ for $l = 1, \ldots, L$. Within each iteration, this results in randomly sampling a smaller sub-network with approximately $\prod_{l=0}^{L-1} \zeta_l$ fraction of all the transformation parameters. Only these $\prod_{l=1}^{L} \zeta_l$ fraction of weights are updated in the current iteration, while the remainder are not. Then, the re-sampling and updating process is repeated. We draw attention to the notation here: $\tilde{\zeta}$ denotes the denoising rate, while $\zeta_0, \ldots, \zeta_{L-1}$ denote the dropout rate.

1. A L-NN may be pretrained layer-wise before supervised tuning (Bengio, 2009a; Vincent et al., 2010; Erhan et al., 2010b). Here, the $L-1$ hidden layers

are first pretrained, for instance, using the box-constrained DA from (5.5). This gives the estimates of the $L-1$ transformations $\mathbf{W}^1, \ldots, \mathbf{W}^{L-1}$, which together with the $\mathbf{y}$s are then used to *initialize* the L-NN. This 'pretrained' L-NN is then learned using dropout as in (5.6). This is the classical regime (Bengio, 2009a; Erhan et al., 2010b), and we discuss this case separately from the fully-supervised version which has *no* layer-wise pretraining. Several studies have already shown interesting empirical relationships between dropout and the DA (Wager et al., 2013), and we complement this body of work by providing explicit relationships between them.

2. 1**-NN with dropout**:
   Given ζ_0, the loss function for the dropout version of 1-NN is

$$\mathcal{L}(\mathbf{x}, \mathbf{y}; \mathbf{W}) = \|\mathbf{y} - \sigma(\mathbf{W}(\mathbf{x} \circ \mathbf{z}^0)\|^2 \tag{5.7}$$

   Although the loss here is a function of the dropout rate ζ^0, we avoid including this in the notation to reduce clutter. The two cases of 1-NN with and without dropout are considered separately in our analysis.

Observe that backpropagation (stochastic gradients on deep networks) proceeds sequentially from the last layer to the input (or first) one updating the L different transformations one at a time (Bottou, 2010). The stepsizes used in the l^{th} layer to update $\mathbf{W}^l$ is denoted by γ^l, and, we use $\mathbf{W}^{k,l}$ to represent the k^{th} gradient update of the l^{th} layer transformation $\mathbf{W}^l$. To keep the presentation simple, while discussing L-NN, whenever appropriate, we refer to the set of unknowns $\{\mathbf{W}^1, \ldots, \mathbf{W}^L\}$ simply as $\mathbf{W}$.

Remark: These three classes broadly encompass both the classical regime where the network is pretrained first using unlabeled data followed by supervised fine tuning (Vincent et al., 2010; Erhan et al., 2010b, 2009), and the more recent fully supervised dropout learning (Srivastava et al., 2014; Krizhevsky et al., 2012). In general, these classes include any deep network trained using noise injection schemes (Matsuoka, 1992; Bishop, 1995; Rifai et al., 2011). Further, the framework presented here is not specific to any specific choice for the non-linearity $\sigma(\cdot)$. Nevertheless, we derive results using the sigmoid non-linearity, $\sigma(x) = 1/(1+e^{-x})$, as a running example because of its simplicity. The results derived for L-NN are then used to adapt the framework to the more popular convolutional and recurrent networks with more sophisticated non-linearities including rectified linear units (ReLUs) (Nair and Hinton, 2010) or

max-outs (Goodfellow et al., 2013) that have been widely used in computer vision. While discussing these complex networks, we show that our analysis and the resulting implications, are applicable broadly with very minimal assumptions on the non-linearities. Note that feature denoising based networks are used only as a starting point for our analysis, instead of the more complicated convolutional networks.

Roadmap

The gradient update in (5.3) is central to the ideas described in this paper. By "tracking" the behavior of these gradients as they propagate across multiple layers of the network, with minimal assumptions on the loss function $\mathcal{L}(\eta; \mathbf{W})$, we can assess the convergence of the overall parameter estimation scheme while taking into account the influence (and structure) of each of the layers involved. Since the updates are stochastic, ideally, we are interested in the "expected" gradients over a certain number of iterations, fixed ahead of time. Motivated by this intuition, our main idea adapted from (Ghadimi and Lan, 2013), is to randomly sample the number of gradient update iterations. Specifically, let N denote the maximum possible number of iterations that can be performed keeping in mind the memory and time constraints (in general, N is very large). The stopping distribution $\mathbb{P}_R(\cdot)$ gives the probability that k^{th} ($k = 1, \ldots, N$) iteration is the *last or stopping* iteration. We denote this randomly sampled stopping iteration as $R \in \{1, \ldots, N\}$, and so,

$$\mathbb{P}_R(\cdot) := \frac{p_R^k}{\sum_{k=1}^{N} p_R^k} \quad \text{where} \quad p_R^k = Pr(R = k) \quad ; \quad k = 1, \ldots, N \tag{5.8}$$

$\mathbb{P}_R(\cdot)$ can either be fixed ahead of time or learned from a hyper-training procedure. An alternate way to interpret the stopping distribution is by observing that p_R^k represents the probability that the estimates ($\mathbf{W}$'s) at the k^{th} iteration are the desired final solutions returned by the learning procedure. This interpretation will be used in the deriving the results. By proceeding with this *random stopping mini-batch stochastic gradients*, and using some Lipschitz properties of the objective and certain distributional characteristics of the input data, we can analyze the three loss functions in (5.4), (5.5) and (5.6). The stopping iteration R is random and the loss function in (5.1) includes an expectation over data instances $\eta = \{\mathbf{x}, \mathbf{y}\}$. Therefore, we are interested in the expectation of the gradients $\nabla_{\mathbf{W}} f(\mathbf{W}^k)$ computed over $R \sim \mathbb{P}_R(\cdot)$ and $\eta \sim \mathcal{X}$. This seemingly small variation to the standard backpropagation leads to gradient convergence

bounds that incorporate the network depth and lengths, and other learning choices like dropout or denoising rate in a very natural manner.

An important hyper-parameter of interest is the variance of $G(\eta; \mathbf{W})$ in (5.3), which in turn depends on the number of free parameters in the given network. This is denoted by e^s, e^{da} and e^m for single-layer, pretraining and multi-layer cases respectively with appropriate subscripts. Beyond these parameters, the distributional hyper-parameters including μ_x and τ_x, and the denoising/dropout rate ζ along with the box-constraint also play a role. Apart from the convergence bounds, we also derive sample sizes required for large deviation estimates of $\mathbf{W}$'s so as to marginalize the influence of the random stopping iteration. This estimate directly relates the dataset size to the number of backpropagation training epochs, where one epoch represents using each training instance, at most once, to compute the mini-batch gradient update from (5.3). This leads to bounds on training time and the minimum required training dataset size, which in turn can be used to ensure a certain level of generalization using existing results (Hardt et al., 2015). Our treatment for the simple case of single-layer networks is presented first, which serves as a basis for the more general settings. The proofs for all the results are included in the appendix.

Why gradient norm? One may ask if characterizing the behavior of the gradients is ideal or if we can do better in terms of characterizing the interplay more directly. There are more than a few reasons why this strategy is at least sensible. First, note that it is NP-hard to check local minima even for simple non-convex problems (Murty and Kabadi, 1987) — so, an analysis using the norms of the gradients is an attractive alternative, especially, if it leads to a similar main result. Second, a direct way of approaching our central question of the architecture and convergence interplay is by analyzing the gradients themselves. Clearly, learnability of the network will be governed by the goodness of the estimates, which in turn depend on the convergence of stochastic gradients. In most cases, this naturally requires an asymptotic analysis of the norm, which, for nonconvex objectives with minimal assumptions (like Lipschitz continuity) was unavailable until very recently (Ghadimi and Lan, 2013). Third, working with the gradient norm directly allows for assessing faster convergence times, which, as argued in (Hardt et al., 2015), is vital for better generalization. Lastly, *post-hoc* checks about the local optimality of the solution provides some empirical evidence supporting the use of gradient norms as will be presented in

Section 5.7.

5.3 Single-layer Networks

Consider a single-layer neural network (1-NN) with visible ($\mathbf{x}$) and output ($\mathbf{y}$) units alone (and no hidden layers). Recall the objective $f(\cdot)$ from (5.1) and the corresponding loss function from (5.4). We learn this 1-NN via a random stopping mini-batch stochastic gradient scheme. The batch size is denoted by B and the randomly chosen stopping iteration R is sampled from the given $\mathbb{P}_R(k)$ ($k = 1, \ldots, N$). This learning procedure is summarized in Algorithm 2; which we refer to as single-layer randomized stochastic gradients, *single-layer RSG*. The special case where $\zeta = 1$ corresponds to using no dropout in the input layer. Recall that $\mathcal{X}$ denotes the training dataset, d_x and d_y are the input and output layer lengths, and $\circ$ in Algorithm 2 refers to element-wise product. $\mathbf{W}^1$ denotes the estimate/initialization at the first iteration, $\mathbf{W}^R$ is the final estimate and γ^k represents the stepsize at the k^{th} iteration. Our first set of results correspond to this single-layer RSG from Algorithm 2.

Algorithm 2 Single-layer Randomized Stochastic Gradients (Single-layer RSG)

Input: d_x, d_y, B, N, γ^k, $\mathbb{P}_R(\cdot)$, $\mathcal{X}$, $\mathbf{W}^1$
Output: $\mathbf{W}^R \in \mathbb{R}^{d_y \times d_x}$

$R \sim \mathbb{P}_R(\cdot) \quad z^1, \ldots, z^{d_x} = 1 \quad \mathcal{J} = \mathbf{1}_{d_h \times d_v}$
for $k = 1, \ldots, R-1$ **do**
  $\{\mathbf{x}^i, \mathbf{y}^i\} \sim \mathcal{X}$, $i = 1, \ldots, B$
  if dropout **then**
    $z_1, \ldots, z_{d_x} \sim \text{Bernoulli}(\zeta^0) \quad \mathcal{J}_{\cdot,j} = 0 \quad \forall\, j \in 1, \ldots, d_v; z_j = 0\}$
  end if
  $\mathbf{x}^i \leftarrow \mathbf{x}^i \circ \mathbf{z}$, $\eta^i := \{\mathbf{x}^i, \mathbf{y}^i\}$
  $\mathbf{W}^{k+1} \leftarrow \mathbf{W}^k - \mathcal{J} \circ \left(\frac{\gamma^k}{B} \sum_{i=1}^{B} \nabla_{\mathbf{W}} \mathcal{L}(\eta^i; \mathbf{W}^k)\right)$ where $\mathcal{L}(\eta; \mathbf{W})$ is from (5.7)
end for

With no prior information about how to setup the stopping distribution, we may want to simply choose R uniformly between 1 and N i.e., $\mathbb{P}_R(k) := \mathtt{Unif}[1, N]$. The first result summarizes the decay of the expected gradients for this setting. Let $D_f = f(\mathbf{W}^1) - f^*$ denote the initial deviation of the objective from unknown optimum $\mathbf{W}^*$.

Theorem 5.1 (**Single-layer Network with Constant Stepsize**). *Consider a single-layer RSG with no dropout ($\zeta^0 = 1$) from Algorithm 2 and constant stepsize $\gamma^k = \gamma \; \forall \; k$. Let $e_\gamma^s = (1 - \frac{13}{16}\gamma)$ and $R \sim \mathtt{Unif}[1, N]$. The expected gradients are given by*

$$\mathbb{E}_R(\|\nabla_{\mathbf{W}} f(\mathbf{W}^R)\|^2) \leqslant \frac{1}{e_\gamma^s}\left(\frac{D_f}{N\gamma} + \frac{e^s\gamma}{B}\right) \tag{5.9}$$

and the optimal constant stepsize is $\gamma_o = \sqrt{\frac{B f(\mathbf{W}^1)}{e^s N}}$ where $e^s = \frac{13 d_x d_y}{256}$.

Remarks: We point out that the *asymptotic* behavior of gradients in backpropagation, including variants with adaptive stepsizes, momentum etc. has been well studied (Magoulas et al., 1999; Becker and Le Cun, 1988; Castillo et al., 2006; Shao and Zheng, 2011). This aspect is *not* novel to our work, as discussed in Section 5.1. However, to our knowledge, relatively few *explicit* results about the convergence *rates* are known; although imposing restrictions on the objective does lead to improved guarantees (Ahmad et al., 1990). Part of the reason may be the lack of such results in the numerical optimization literature for general nonconvex objectives. The recent result in (Ghadimi and Lan, 2013) presents one of the first such results addressing general nonconvex objectives with a first-order oracle. Consequently, we believe that Theorem 5.1 gives non-trivial information and, as we will show in later sections, leads to new results in the context of neural networks. This result serves as a building block for analyzing feature denoising and multi-layer dropout later in Section 5.4 and 5.5. We now explain (5.9) briefly.

While the first term corresponds to the goodness of fit of the network (showing the influence of the deviation D_f), the second term encodes the degrees of freedom of the network (via e^s), and is larger for big (or fatter) networks. Clearly, the bound decreases as N (and/or B) increase, and it is interesting to see that the effect of network size ($d_x d_y$) is negligible for large batch sizes. Note that the second term of the bound induces a kind of bias, which depends on the variance of the noisy gradients e^s and batchsize B. As B increases, the gradient update is averaged over a large number of samples, and (5.3) will be much closer to a full-batch gradient update, a classical property of mini-batch stochastic gradients (LeCun et al., 2012; Bengio, 2012). The constant e_γ^s depends on the stepsize and increases the overall bound as γ increases. The second term in the bound suggests that larger batch sizes are necessary for fatter networks with large d_x or d_y. One caveat of the generality of (5.9) is that it might be loose in the worst case where $D_f \sim 0$ (i.e., when $\mathbf{W}^1$ is already a good estimate

of the stationary point). The optimal constant stepsize γ_o in Theorem 5.1 is calculated by balancing the two terms in (5.9). Using this γ_o, it is easy to see that the expected gradients have the following rates.

Corollary 5.2 (**Rates of Single-layer Network**). *For a single-layer RSG with no dropout from Algorithm 2 with optimal constant stepsize γ_o from Theorem 5.1, we have*

$$\mathbb{E}_R(\|\nabla_{\mathbf{W}} f(\mathbf{W}^R)\|^2) := \begin{cases} \mathcal{O}\left(\frac{1}{\sqrt{N}}\right) & \textit{for a given network} \\ \mathcal{O}(\sqrt{d_x d_y}) & \textit{for a given } N \end{cases} \tag{5.10}$$

An interesting relationship between specifying constant stepsizes and choosing a uniform stopping distribution can be seen from the proof of Theorem 5.1. Specifically, using constant stepsizes directly corresponds to choosing a uniform stopping distribution i.e., they are *equivalent* in some sense. One can nevertheless use any $\mathbb{P}_R(\cdot)$ with constant stepsizes. Clearly, whenever N is very large, one may want to allow the stopping iteration R to be as close to N as possible – a uniform $\mathbb{P}_R(\cdot)$ does not necessarily allow that. Such alternate $\mathbb{P}_R(\cdot)$s in tandem with constant stepsizes will be discussed shortly in Corollary 5.4, where we show that they lead to a different set of constants in (5.9), but the high-level dependence on the hyper-parameters including N and the network size $d_x d_y$ will remain the same. Before analyzing these general stopping distributions, we first address another common setting in stochastic gradients where decaying γ^ks (as k increases) are used. Theorem 5.3 summarizes the decay of the expected gradients using such monotonically decreasing stepsizes $\gamma^k = \frac{\gamma}{k^\rho}$ for some given $\rho > 0$. Here, $\mathcal{H}_N(\cdot)$ is the generalized harmonic number defined as $\mathcal{H}_N(\theta) = \sum_{i=1}^{N} \frac{1}{i^\theta}$.

Corollary 5.3 (**Single-layer Network with Decreasing Stepsize**). *Consider a single-layer RSG with no dropout from Algorithm 2 and stepsizes $\gamma^k = \frac{\gamma}{k^\rho}$. Let the probability of stopping at the k^{th} iteration $p_R^k = \gamma^k(1 - \frac{13}{16}\gamma^k)$. The expected gradients are given by*

$$\mathbb{E}_R(\|\nabla_{\mathbf{W}} f(\mathbf{W}^R)\|^2) \leqslant \frac{16}{3\mathcal{H}_N(\rho)} \left(\frac{D_f}{\gamma} + \frac{e^s \gamma \mathcal{H}_N(2\rho)}{B} \right) \tag{5.11}$$

and the optimal γ is $\sqrt{\frac{B f(\mathbf{W}^1)}{e^s \mathcal{H}_N(2\rho)}}$ where $e^s = \frac{13 d_x d_y}{256}$

Remarks: The influence of N on the bound in (5.11) is via $\mathcal{H}_N(\rho)$. The interaction of ρ with N and other hyper-parameters, and its eventual dependence on

expected gradients decay is complicated. Nevertheless, broadly, as ρ increases, the bound first decreases and eventually increases becoming looser. This trend is expected and not surprising. First observe that, whenever $\rho \approx 0$, the stepsizes are approximately constant (i.e, $\gamma^k \approx \gamma, \forall k$). Here, we have $\mathcal{H}_N(\rho) \approx N$, and assuming that $\gamma < 1$, the bound in (5.11) is at least as large as the bound from (5.9) making Corollary 5.3 consistent with Theorem 5.1. Alternatively, large ρ implies strong decay of the stepsizes. For sufficiently small γ, this results in a stopping iteration probability p_R^k that decreases at a fast rate as k increases (see its definition from Corollary 5.3), i.e., large ρ results in $R \ll N$. Hence, the bound is simply implying that the expected gradients are going to be large whenever the gradient updating is stopped early. For a given ρ, the influence of network size and B is the same as discussed earlier for Theorem 5.1. Overall, these observations clearly imply that the bounds from (5.9) and (5.11) are capturing all the intuitive trends one would expect to see from the use of stochastic gradients, in turn, making the analysis and results more useful.

Observe that Corollary 5.3 enforces a specific structure on $\mathbb{P}_R(\cdot)$, and $R \sim \texttt{Unif}[1, N]$ (from Corollary 5.1) is restrictive, in the sense that, one may want to choose R as close to N as possible. The following result shows the decay of expected gradients whenever $\mathbb{P}_R(\cdot)$ is monotonically *increasing*, i.e., $p_R^k \leqslant p_R^{k+1}, \forall k$, thereby pushing R towards N with high probability.

Corollary 5.4 (**Single-layer Network with Monotonic** $\mathbb{P}_R(\cdot)$)**.** *Consider a single-layer RSG with no dropout from Algorithm 2 and constant stepsize* $\gamma^k = \gamma \forall k$*. Let* $e_\gamma^s = (1 - \frac{13}{16}\gamma)$ *and* $p_R^k \leqslant p_R^{k+1} \forall k = 1, \ldots, N$*. The expected gradients are given by*

$$\mathbb{E}_{R,\eta}(\|\nabla_{\mathbf{W}} f(\mathbf{W}^R)\|^2) \leqslant \frac{p_R^N}{\gamma e_\gamma^s}\left(D_f + \frac{e^s N}{B}\gamma^2\right) \tag{5.12}$$

and the optimal constant stepsize is $\sqrt{\frac{B f(\mathbf{W}^1)}{e^s N}}$ *where* $e^s = \frac{13 d_x d_y}{256}$

Remarks: The uniform stopping distribution is a special case with $p_R^N = 1/N$, where the bound in (5.12) reduces to (5.9). Clearly, in the extreme case where $R = N$ with probability approaching one, (5.12) is loose. One can ensure that $R \approx N$ while the bound is reasonably tight by choosing $\mathbb{P}_R(\cdot)$ to have more mass on the last few iterations. For instance, consider the following stopping distribution for some given $\vartheta > 1$.

$$p_R^k = \begin{cases} 0 & \text{for} \quad k = 1, \dots, N(1 - \frac{1}{\vartheta}) \\ \frac{\vartheta}{N} & \text{for} \quad k = 1 + N(1 - \frac{1}{\vartheta}), \dots, N \end{cases} \tag{5.13}$$

Using this specification and (5.12), we then have

$$\mathbb{E}_{R,\eta}(\|\nabla_{\mathbf{W}} f(\mathbf{W}^R)\|^2) \leqslant \frac{\vartheta}{e_\gamma^s} \left(\frac{D_f}{N\gamma} + \frac{e^s \gamma}{B} \right) \tag{5.14}$$

In (5.14), ϑ may be chosen so that $\mathbb{P}_R(\cdot)$ has non-zero mass over the last few hundred iterations. For a given N and B, the bound in (5.14) can be made as tight as possible by appropriately choosing ϑ, thereby ensuring convergence of expected gradients of Algorithm 2, while forcing R to be as large as possible (unlike the setting from Theorem 5.1). The cost of enforcing such a $\mathbb{P}_R(\cdot)$ (like (5.13)) is only a slight modification over the typical stochastic gradients used to learn deep networks in practice. The interplay of ϑ, B and N on the network size $d_x d_y$ is nevertheless complex, and we visualize such trends in Section 5.7. Further, the broad structure of (5.12) is similar to those from (5.9) and (5.11), and so, the inference we draw about the influence of B and network size still apply here. Non-constant step sizes and complex stopping distributions are interesting from the modeling perspective, but, to ensure less notational clutter from constants while retaining the broad trends, we focus more on the constant stepsize setting (with uniform and monotonic $\mathbb{P}_R(\cdot)$) for the DA based pretraining and dropout L-NN analysis in the later sections.

Choosing $\mathbb{P}_R(\cdot)$***:*** It is evident that the convergence is sensitive to the choice of $\mathbb{P}_R(\cdot)$, and Theorem 5.1, Corollaries 5.3 and 5.4 cover the *typical* choices of stopping criteria. From a practical stand-point, one can instead use multiple $\mathbb{P}_R(\cdot)$s from a pre-defined dictionary of distributions $\mathcal{P}$, and choose the best one via cross-validation on some reasonable performance measure. For instance, the best $\mathbb{P}_R(\cdot) \in \mathcal{P}$ can be selected based on a validation dataset either by directly computing the empirical average of the gradients, or using alternate measures like generalization performance. This is a valid operation because several recent results justify using the number of training iterations as a 'surrogate' for the generalization performance (Hardt et al., 2015). Further, the analysis also allows probing the nature of R after fixing $\mathbb{P}_R(\cdot)$. Specifically, depending on the Hessian at the R^{th} iteration, gradient updates can be continued for a few more iterations. The results presented here will still hold for this post-hoc increase

in iterations. The visualization of trends from our results (see Section 5.7) and additionally in Section 5.8 provide more insights into these choices from an empirical perspective, including the issue of relating convergence iterations to *expected* generalization. Observe that our statements about $\mathbb{P}_R(\cdot)$ hold for other pretraining and multi-layer networks including the more complex convolutional and recurrent networks, and so, we focus more on the architecture and other hyper-parameters when discussing these complex networks.

Theorems 5.1, 5.3 and 5.4 describe convergence of a single-layer network for *one run* of stochastic gradient. In practice, we are more interested in a large deviation bound over multiple runs, especially because of the randomization over η and R (see (5.9)). We define such a large deviation estimate, using $\mathbf{W}^{R_1}, \ldots, \mathbf{W}^{R_T}$ computed from $T > 1$ *independent* runs of single-layer RSG, and calculate the minimum N required to achieve such an estimate.

Definition 5.5 (**(ϵ, δ)-solution**). *Given $\epsilon > 0$ and $0 < \delta \ll 1$, an (ϵ, δ)-solution of a single-layer network is given by* $\arg\min_t \|\nabla_{\mathbf{W}} f(\mathbf{W}^{R_t})\|^2$ *such that* $\Pr\left(\min_t \|\nabla_{\mathbf{W}} f(\mathbf{W}^{R_t})\|^2 \leqslant \epsilon\right) \geqslant 1 - \delta$.

We can then state the following result for the single-layer RSG with *no* dropout using constant stepsize (the settings from Theorems 5.1 and 5.4).

Theorem 5.6 (**Single-layer RSG Computational Complexity**). *Consider a single-layer RSG with no dropout. Let $\bar{\delta} = \frac{13B\epsilon\delta^{1/T}}{32e^s}$. To compute a (ϵ, δ)-solution with $\gamma^k = \gamma$ and monotonic stopping distribution $p_R^k \leqslant p_R^{k+1} \forall k$ with $p_R^N = \frac{\vartheta}{N}$, Algorithm 2 needs*

$$N(\epsilon, \delta) \geqslant \frac{4e^s f(\mathbf{W}^1)}{B\delta^{2/T}\epsilon^2} \left(\bar{\delta} + \vartheta\right)^2 \tag{5.15}$$

Remarks: For the special case of uniform stopping distribution, ϑ would simply be 1. To illustrate the practical usefulness of this result, consider an example network with $d_x = 100$, $d_y = 5$ and say $f(\mathbf{W}^1) \approx d_y$. To compute a $(0.05, 0.05)$-solution with $T = 10$ runs and a batchsize $B = 50$, (5.15) gives $N > 7.8 \times 10^3$. If the number of epochs is $C = 200$, then under the (reasonable) assumption that $SC \approx BN$ (where S is sample size), the aforementioned result asks for ~ 2000 instances. Also, (5.15) implies that N increases as the network size ($d_x d_y$) and the initial deviation ($f(\mathbf{W}^1)$) increase. Similarly, for obtaining good large deviation estimates (i.e., very small ϵ and δ), the number of required iterations is large. Corollary 5.6 does not use any information about the input data statistics (e.g., moments), and so expectedly, (5.15) may overestimate N,

and therefore S. Later, we make use of such data statistics to analyze networks with hidden layers, while the rest of the procedure will generalize what is presented here (for the single-layer setting).

Single-layer RSG with Dropout: Recall the dropout induced single-layer network from (5.7). In each iteration, the input units are dropped based on a given dropout rate ζ, i.e., the transformation parameters for these dropped units are not updated. The following result summarizes the expected gradients in this setting, where a random subset of $(1-\zeta)d_x d_y$ parameters (of all the $d_x d_y$ unknowns) are updated in each iteration.

Corollary 5.7 (Single-layer Network with Dropout). *Consider a single-layer RSG with input dropout rate ζ from Algorithm 2 and constant stepsize $\gamma^k = \gamma \forall k$. Let $e_\gamma^s = (1-\frac{13}{16}\gamma)$, $e^s = \frac{13 d_x d_y}{256}$, $\bar{\delta} = \frac{13 B \epsilon \delta^{1/T}}{32 e^s}$ and $R \sim \mathtt{Unif}[1, N]$. The expected gradients are given by*

$$\begin{aligned} \mathbb{E}_R(\|\nabla_{\mathbf{W}} f(\mathbf{W}^R)\|^2) &\leqslant \frac{1}{e_\gamma^s}\left(\frac{D_f}{N\gamma\zeta} + \frac{e^s\gamma}{B}\right) \\ N(\epsilon,\delta,\zeta) &\geqslant \frac{4 f(\mathbf{W}^1) e^s}{\zeta B \delta^{2/T} \epsilon^2}(1+\bar{\delta})^2 \end{aligned} \tag{5.16}$$

with optimal constant stepsize $\sqrt{\frac{B f(\mathbf{W}^1)}{\zeta e^s N}}$

Remarks: Since approximately $\zeta d_x d_y$ unknowns are not updated in each gradient iteration, the dropout operation induces noise into the loss function (see (5.7)). This results in a solution path that is a noisy approximation of the non-dropout version. Hence, the stochastic gradients procedure needs to work 'harder' in some sense, to optimize the loss: (5.16) shows this behavior via $1/\zeta$ in the first term. Correspondingly, in comparison to (5.15) (from Theorem 5.6), many more gradient iterations i.e., larger N, are required to compute the same (ϵ, δ) solution. In the extreme case where $\zeta \approx 0$, the bound from (5.16) is loose. Although the single-layer dropout is simple from the perspective of generalization, this setup, and the above result, are key to multi-layer networks in Section 5.5 where we analyze dropout in L-NNs.

5.4 Unsupervised Pretraining

Building upon Theorems 5.1–5.7, we now consider single-layer networks that perform unsupervised learning. One widely used pretraining scheme is ana-

lyzed in detail here – feature denoising based autoencoders (DA) (Vincent et al., 2010). As discussed in Section 5.2, these schemes belong to to a broader family of input/feature corruption based regularization procedures (Matsuoka, 1992; Bishop, 1995), which have been widely used for layer-wise pretraining of deep networks. The setting we consider is a box-constrained DA (see the loss function from (5.5)). The random stopping mini-batch stochastic gradients learning for this box-constrained DA is summarized in Algorithm 3. $P_{\mathbf{W}}$ denotes the Euclidean projection operator onto the constraint set $[-w_m, w_m]^{d_h \times d_x}$. Unlike the single-layer setting, because of the constraint on $\mathbf{W}$, our interest now is in the expected *projected* gradients denoted by $\nabla_{\mathbf{W}} \tilde{f}(\mathbf{W})$, which simply correspond to the Euclidean projection of $\nabla_{\mathbf{W}} f(\mathbf{W})$ on $\mathbf{W} \in [-w_m, w_m]^{d_h \times d_x}$.

Algorithm 3 Box-constrained Denoising autoencoder Randomized Stochastic Gradients (DA RSG)

Input: d_x, d_h, B, N, γ^k, $\tilde{\zeta}$, w_m, $\mathbf{W}^1$, $\mathbb{P}_R(\cdot)$
Output: $\mathbf{W}^R \in \mathbb{R}^{d_h \times d_x}$

$R \sim \mathbb{P}_R(\cdot)$;
for $k = 1, \ldots R-1$ **do**
  $z_1^i, \ldots, z_{d_x}^i \sim$ Bernoulli$(\tilde{\zeta})$
  $\mathbf{x}^i \sim \mathcal{X}$, $\eta^i := \{\mathbf{x}^i \circ \mathbf{z}^i\}$, $i = 1, \ldots, B$
  $\mathbf{W}^{k+1} \leftarrow P_{\mathbf{W}}(\mathbf{W}^k - \frac{\gamma^k}{B} \sum_{i=1}^{B} \nabla_{\mathbf{W}} \mathcal{L}(\eta^i; \mathbf{W}))$ where $\mathcal{L}(\eta; \mathbf{W})$ is from (5.5)
end for

As mentioned in the remarks of Corollary 5.4, we restrict the discussion to constant stepsizes, first for the uniform stopping distribution and later for the more general one, to simplify our discussion, while noting that the trends remain the same even with alternate stepsizes and choices of $\mathbb{P}_R(\cdot)$. The following result bounds $\nabla_{\mathbf{W}} \tilde{f}(\mathbf{W})$ for DA RSG procedure from Algorithm 3. D_f denotes the initial deviation from the optimum, while e^{da} and e_γ^{da} are the hyper-parameters representing the network structure and learning constants.

Theorem 5.8 (DA with constant stepsize). *Consider the DA RSG with* $\mathbf{W} \in [-w_m, w_m]^{d_h \times d_x}$. *Let*

$$e^{da} = \frac{d_x d_h}{16}\left[1 + \frac{\tilde{\zeta} d_x w_m}{4}\mu_x + \left(\frac{5\tilde{\zeta}}{16} - \frac{\tilde{\zeta}^2}{4}\right)(\tilde{\zeta} d_x w_m)^2 \tau_x\right] \tag{5.17}$$

and $\mathcal{U}_{da}$ *denote the Lipschitz constant of* $\nabla_{\mathbf{W}} f(\mathbf{W})$ *for the loss from* (5.5). *Using constant stepsize* $\gamma < \frac{2}{\mathcal{U}_{da}}$ *with* $R \sim \mathtt{Unif}[1, N]$ *and denoting* $e_\gamma^{da} = 1 - \frac{\mathcal{U}_{da}}{2}\gamma$, *the expected projected gradients is*

$$\mathbb{E}_R(\|\nabla_{\mathbf{W}}\tilde{f}(\mathbf{W}^R)\|^2) \leqslant \frac{D_f}{N\gamma e_\gamma^{da}} + \frac{e^{da}}{B}\left(1 + \frac{1}{e_\gamma^{da}}\right) \tag{5.18}$$

and the optimal stepsize is $\sqrt{\frac{2Bf(\mathbf{W}^1)}{\mathcal{U}_{da}e^{da}N}}$.

Remarks: Similar to the bound from (5.9) for the single-layer RSG, the above result in (5.18) combines the contributions from the output goodness of fit (D_f), the number of free parameters ($d_h d_x$) and the stepsize choice (γ, e_γ^{da}). Here, D_f is balanced out by the number of iterations N in the first term. The second term involving the variance of gradients (e^{da}) and the batchsize B controls the 'bias' from the network's degrees of freedom – the remarks from Theorem 5.1 still apply here. However, here we find that the dependence on the network structure is more involved. The input and hidden layer lengths do not contribute equally (see e^{da} from (5.17)) which can be partly explained by the structure of the loss function (see (5.5)). For smaller constraint sets i.e., small w_m, and therefore small e^{da}, we expect the projected gradients to typically have small magnitude, which is clearly implied by (5.18). In practice, w_m is reasonably small, and, in general, several studies have shown that W_{ij}'s tend to emulate a 'fat' Gaussian distribution with mean approximately around zero (Bellido and Fiesler, 1993; Blundell et al., 2015). All the interpretations about the influence of stopping distributions, and the viable choices to ensure faster convergence, that were discussed in Section 5.3 in the context of single-layer RSG still hold true for DA RSG. Hence, we focus more on the DA-specific hyper-parameters including the data moments.

From Denoising Rates and Network structure to (5.18)

We see that (5.17) and its resulting structure in the bound from (5.18) seems to imply a non-trivial interplay between the network size $d_h d_x$, the data statistics (via μ_x and τ_x where $\mu_x = \frac{1}{d_x}\sum_j \mathbb{E}x_j$ and $\tau_x = \frac{1}{d_x}\sum_j \mathbb{E}^2 x_j$) and the denoising rate $\tilde{\zeta}$. A direct observation from the bound in (5.18) is that it is always useful to use smaller (or thinner) networks, which results in faster decay of expected gradients as iterations increase. This trend, in tandem with observations from (Hardt et al., 2015) about generalization imply the superiority of thinner networks compared to fatter ones – (Romero et al., 2014) and others have already shown evidence of this behavior. As was observed with single-layer networks from (5.9), fatter DAs will need a large batchsize B. We further discuss this

trade-off of thinner versus fatter networks later in Section 5.5 when presenting multi-layer networks (sections 5.7 and 5.7 show some interesting plots summarizing the interplay from (5.18)). Beyond these visualizations and evaluations, we intend to interpret this interaction and its corresponding influence on the decay of (5.18) for a few of the commonly encountered cases. Here, we assume that the batchsize B and the maximum allowable iterations N are reasonably large.

Small data moments, $\mu_x \approx 0$, $\tau_x \approx 0$: In this trivial case, the signal is very weak and there may not be much to learn. Clearly, e^{da} will be close to its minimum which will directly result in faster convergence in terms of the expected projected gradients. Recall that smaller $\tilde{\zeta}$ implies more noisy gradients (across multiple iterations). However, $\tilde{\zeta}$'s influence on the bound is nullified because $\mu_x, \tau_x \approx 0$. This implies that independent of the complexity of the task being learned, there is no reason for using large denoising rates whenever the average of input across all features is small. This is sensible because the small irregularities, in the form of noise or dampening of inputs, which would be induced by the DA corruption process might in fact correspond to class-specific changes in the data. So, using a smaller $\tilde{\zeta}$ will require many more gradient iterations to convergence to some solution (as implied by (5.18)), while also resulting in poor hidden representations. With larger $\tilde{\zeta}$, the only contributing factor for convergence is the network size.

Small denoising rate, $0 \ll \tilde{\zeta} < 1$: When $\tilde{\zeta}$ is large, the input corruption is very small and $\mathbf{x} \circ \mathbf{z} \approx \mathbf{x}$ (refer to the loss from (5.5), $\circ$ denotes element-wise product). Here, the noise in the gradients $\nabla_{\mathbf{W}} \mathcal{L}(\eta; \mathbf{W})$ is almost entirely governed by the data statistics. Within this setting,

1. *Small data moments:* Small μ_x and τ_x can lead to faster convergence, and as they increase, the bound becomes loose. The trend here is similar to small data moments case discussed earlier.
2. *Large data moments:* When μ_x and τ_x are reasonably large however, the bound can be controlled only if stepsize is small enough, with a large batchsize and reasonably small lengths of inputs. This setting is closer to the classical non-regularized autoencoder, and hence, the representations learned might not be useful – (5.18) predicts this same behavior, albeit from a parameter convergence perspective. To see this, observe that faster convergence time is desired for better generalization (Hardt et al., 2015). Such faster times cannot be guaranteed, here, whenever the network is reasonably large (even with large d_x and small d_y) with large

data moments, because the number of gradient iterations will need to be correspondingly large according to (5.18). Clearly, if the convergence bound is forced to be very small, the resulting representations are almost surely overfitting the inputs. Note that in the pathological case where $\mu_x \approx 1$, $\tau_x \approx 1$ with $\tilde{\zeta} \approx 1$, the bounds are too loose to be relevant.

Large denoising rate, $0 < \tilde{\zeta} \ll 1$:

In the opposite scenario, with small a $\tilde{\zeta}$, we see that e^{da} is going to be close to $\frac{d_x d_h}{16}$.

1. *Not-so-large data moments:* Whenever the data moments are not too large, their influence is almost surely mitigated because of the presence of large corruptions induced by small $\tilde{\zeta}$ (see (5.5)). This is the 'good' DA regime – where the aim is to perturb the data without necessarily destroying *all* the class-specific information. As suggested by the interplay in (5.18), there might be good combinations of d_h, B, γ and N that will lead to good decay of expected gradients while also generating very useful representations – these trends have been analyzed empirically in (Vincent et al., 2010; Erhan et al., 2010b).
2. *Large data moments:* If the data moments are quite large (≈ 1), the convergence is almost entirely controlled by $\tilde{\zeta}$, which in turn will be faster for thinner networks with large B (the only other hyper-parameters playing a non-trivial role within this setting). Here, the trends are similar to the large data moments regime discussed earlier with small $\tilde{\zeta}$ mainly because e^{da} cannot be controlled anymore. The worst scenario is when $\mu_x \approx 1$, $\tau_x \approx 1$, in which case the improvement resulting from a small last term in e^{da} will be compensated by the second term, naturally leading to a loose bound.

Size of constraint set, Choice of w_m:

For small w_m, the projected gradients are expected to have small magnitude, which is intuitive because the magnitude of the projected gradients will be bounded by the diameter of the constraint set. Notice that (5.18) also predicts this behavior via the structure of e^{da}, which is quadratic in w_m and decreases as w_m decreases. On the other hand, as one increases the set size w_m, for a given network size and $\tilde{\zeta}$, we are clearly forced to increase N and/or B correspondingly. Since the choice of w_m is independent of data statistics and other hyper-parameters, all the earlier statements (for the three other cases) about the distributional and learning constants including network sizes will come into play as w_m increases.

Beyond these prototypical settings, for small stepsizes γ, e_γ^{da} will be as large as possible making the bound tighter. Clearly, the influence of both data moments and $\tilde{\zeta}$ on the decay of gradients will be mitigated by increasing B and N alone, but, as suggested in (Hardt et al., 2015), this will be at the cost of poorer generalization. The above discussed regimes are some of the widely used ones, but the interplay of all the terms used in Theorem 5.8 and (5.18) is more involved. There are certain situations where (5.18) (and (5.9), as well, from Theorem 5.1) will be loose. For instance, across all regimes, the case of $\mu_x \approx 1$ and $\tau_x \approx 1$ seems to be pathological. In such a case, the data dimensions may be un-informative and highly correlated to begin with. Similarly, as we keep increasing d_x and d_h (or d_y for 1-NN) to very large values, at some point, the bounds will become loose but depending on γ, $\tilde{\zeta}$ and data moments, it may be the case that the learned representations are still meaningful. Further, e_γ^{da} decreases as γ increases, and when the stepsize becomes too large (beyond reasonable values), the resulting bounds are not very informative even if empirically the architecture could still work well. Finally, (5.18) is less useful for the unconstrained case. We note that, our framework does handle this, but the corresponding analysis may not provide any new insights on the interplay of structure and learning beyond what is predicted by Theorem 5.8.

Recall the discussion about faster convergence time versus generalization performance from Section 5.2 and also (Hardt et al., 2015). Using the above *guidelines* one can ensure smallest possible N for good generalization. For completeness, the following result presents the decay of expected gradients for DA RSG using constant stepsize with monotonically increasing $\mathbb{P}_R(\cdot)$ (the setting from Corollary 5.4) with $p_R^k \leqslant p_R^{k+1} \forall k$.

Corollary 5.9 (DA with Monotonic $\mathbb{P}_R(\cdot)$). *Consider the DA RSG with $\mathbf{W} \in [-w_m, w_m]^{d_h \times d_x}$. Let e^{da} be given by* (5.17) *and $\mathcal{U}_{da}$ denote the Lipschitz constant of $\nabla_{\mathbf{W}} f(\mathbf{W})$ with loss from* (5.5)*. Using constant stepsize $\gamma < \frac{2}{\mathcal{U}_{da}}$ with $p_R^k \leqslant p_R^{k+1} \forall k$ and denoting $e_\gamma^{da} = 1 - \frac{\mathcal{U}_{da}}{2}\gamma$, the expected projected gradients is*

$$\mathbb{E}_R(\|\nabla_{\mathbf{W}} \tilde{f}(\mathbf{W}^R)\|^2) \leqslant \frac{p_R^N D_f}{\gamma e_\gamma^{da}} + \frac{e^{da}}{B}\left(1 + \frac{p_R^N N}{e_\gamma^{da}}\right) \tag{5.19}$$

and the optimal stepsize is $\sqrt{\frac{2Bf(\mathbf{W}^1)}{\mathcal{U}_{da} e^{da} N}}$

Remarks: Under the special case where $\mathbb{P}_R(\cdot)$ is uniform, the bound in (5.19) reduces to (5.18). Also as observed earlier with Corollary 5.4, one can ensure that $R \approx N$ in tandem with a tighter (and the smallest possible) bound for the

given N and B by choosing $\mathbb{P}_R(\cdot)$ as in (5.13), where more mass is accounted by the last few iterations. Using (5.13), (5.19) reduces to (recall that $\vartheta > 1$)

$$\mathbb{E}(\|\nabla_{\mathbf{W}}\tilde{f}(\mathbf{W}^R)\|^2) \leqslant \frac{\vartheta D_f}{N\gamma e_\gamma^{da}} + \frac{e^{da}}{B}\left(1 + \frac{\vartheta}{e_\gamma^{da}}\right) \tag{5.20}$$

The cost of enforcing such a $\mathbb{P}_R(\cdot)$ is just a minor modification on DA learning procedures (see Algorithm 3). Several other observations made earlier about the viable and interesting choices of $\mathbb{P}_R(\cdot)$ (see Remarks of Corollary 5.4, and Section 5.3) still hold true for the above result, and hence for noise-injection based autoencoder learning in general.

Finally, the following result presents the computational complexity of DA RSG, similar to Theorem 5.6 which computes the number of gradient iterations N required for estimating a (ϵ, δ) solution (see Definition 5.5) for single-layer RSG. Here, we assume that $SC \approx BN$ where S is the sample size (training dataset size) and C denotes the number of epochs. Note that the same assumption was used in the Remarks of Theorem 5.6 to discuss some example sample sizes.

Theorem 5.10 (**DA RSG computational Complexity**). *Consider a DA RSG learned with constant stepsize γ, denoising rate $\tilde{\zeta}$ and uniform stopping iteration. To compute a (ϵ, δ)-solution, we need*

$$B \geqslant \frac{2e^{da}}{\epsilon\delta^{1/T}}\left(1 + \frac{1}{e_\gamma^{da}}\right) \textit{ and with such } B \;\; N \geqslant \max\left\{\frac{SC}{B}, \frac{2d_x}{\gamma e_\gamma^{da}\epsilon\delta^{1/T}}\right\} \tag{5.21}$$

where C is the number of backpropagation epochs and S is the number of unsupervised training instances available.

Remarks: The above result provides a recipe for setting the batchsize and maximum number of iterations for the DA RSG. As the network size increases, e^{da} increases which in turn implies larger batchsize B, and larger input size d_x increases the maximum iterations N. Observe that $f(\mathbf{W}^1)$ is replaced by d_x there by making (5.21) computable in practice. Once B and N are fixed using (5.21), the T different random stopping iterations corresponding to the T different DA RSGs can be chosen independently. This result will be used in one of the steps of the design procedures presented in Section 5.8.

5.5 Multi-layer Networks

We now extend our analysis to multi-layer networks. Using Theorem 5.8 as a starting point, which analyzes a single hidden layer network using feature

denoising, we first consider a L-layered multi-layer network that performs *layer-wise pretraining before* backpropagation based supervised finetuning. As discussed in Section 5.2 while discussing the L-NN loss in (5.6), this two-fold learning mechanism corresponds to the classical deep learning regime where large amounts of unsupervised data are used to initialize the network transformations. The resulting bound allows us to incorporate feature dropout in both the input and hidden layers during the supervised tuning stage, and eventually leads to our final result on the most general case of dropout learning for L-NNs. A consequence of this general result is a clear and intuitive relation between dropout based supervised learning, layer-wise pretraining and other structural and distributional parameters. Algorithm 4 presents the random stopping stochastic gradients for multi-layer networks, referred to as multi-layer RSG.

Layer-wise Pretraining

The $L-1$ hidden layers are pretrained using box-constrained denoising autoencoders from Algorithm 3, resulting in estimates of the unknowns $\mathbf{W}^1, \ldots, \mathbf{W}^{L-1}$ denoted by $\mathbf{W}^{1,1}, \ldots, \mathbf{W}^{1,L-1}$ (the superscripts $(1, \cdot)$ simply correspond to the first iteration of L-NN learning, see description of L-NNs from Section 5.2). These estimates, along with a random estimate of $\mathbf{W}^L$, denoted by $\mathbf{W}^{1,L}$, will then initialize the L-NN. The final estimates are then learned via supervised back-propagation using $\{\mathbf{y}\}$s and the loss (5.6). For notational convenience we collectively denote the final estimate $\mathbf{W}^{R,1}, \ldots, \mathbf{W}^{R,L}$ simply as $\mathbf{W}^R$ in the results. Since the pretraining stage uses box-constraints, we are still interested in the expected projected gradients accumulated across all the layers. The following result shows the decay of these expected projected gradients. D_f here denotes the initial deviation of the objective from (5.6) *after* the $L-1$ layers have been pretrained. Similar to e^{da} from Theorem 5.8, e_l^ms for $l = 1, \ldots, L$ encode the structural and learning hyper-parameters of the network. We assume that all the hidden layers are pretrained to the same degree i.e., each of them is pretrained to a given (α, δ_α) solution (see Definition 5.5).

Theorem 5.11 (**Multi-layer Network**). *Consider a multi-layer RSG with no dropout from Algorithm 4 learned via box-constrained layer-wise pretraining from Algorithm 3 and supervised backpropagation with constant stepsizes $\gamma^l \forall l$. Let $e_l^m = \frac{\gamma^l}{4} d_{l-1} d_l d_{l+1} w_m^l$ for $l = 1, \ldots, L-1$ and $e_L^m = \frac{13 d_{L-1} d_L \gamma_L^2}{256}$. Whenever $\gamma_l < \frac{20}{\alpha d_{l+1} w_m^l}$, and the hidden layers are pretrained for a (α, δ_α) solution (from Definition 5.5), we have*

Algorithm 4 Multi-layer Randomized Stochastic Gradients (Multi-layer RSG) – Dropout with/without layer-wise pretraining

Input: $d_1, \ldots, d_L$, B, N, $\gamma_1^k, \ldots, \gamma_L^k$, $\mathbb{P}_R(\cdot)$, $\mathcal{X}$, $w_m^1, \ldots, w_m^{L-1}$, $\mathbf{W}^{1,L}$, $\zeta_0, \ldots, \zeta_{L-1}$
Output: $\mathbf{W}^{R,l} \in \mathbb{R}^{d_{l+1} \times d_l}$ for $l = 1, \ldots, L-1$

```
for l = 1, ..., L − 1 do
  if pretrain then
    W^{1,l} ← Algorithm 3
  else
    W^{1,l} ← Random Initialization
  end if
end for
R ~ P_R(·);   J^l = 1_{d_{l+1} × d_l} for l = 1, ..., L − 1
for k = 1, ... R − 1 do
  {x^i, y^i} ~ X
  if dropout then
    z_1^l, ..., z_{d_x}^l ~ Bernoulli(ζ_l)   l = 0, ..., L − 1
    J^l_{·,k} = 0 ∀ k = {i ∈ 1, ..., d_l; z_i^l = 0}
  end if
  x ← x ∗ z^0   h^l ← h^l ∘ z^l
  η^i := {x^i, y^i}
  W^{k+1,L} ← W^{k,L} − J^L ∘ (γ_L^k / B) Σ_{i=1}^B ∇_{W^l} L(η^i; W^{k,l})
  for l = L − 1, ..., 1 do
    ∇^{pq}_{W^l} L(η^i; W^l) = h_q^{l−1} h_p^l (1 − h_p^l) ⟨P^{·p}_{W^{l+1}}(η^i; W^{k+1,l+1}), W^{k+1,l+1}_{·p}⟩
    ∀p,q
    W^{k+1,l} ← P_{W^l}(W^{k,l} − J^l ∘ (γ_l^k / B) Σ_{i=1}^B ∇_{W^l} L(η^i; W^{k,l}))
  end for
end for
```

$$\mathbb{E}_R \|\nabla_{\mathbf{W}} \tilde{f}(\mathbf{W}^R)\|^2 \leqslant \frac{1}{e_\gamma^m} \left(\frac{D_f}{N} + \frac{1}{B} (e_L^m + \alpha \sum_{l=1}^{L-1} e_l^m) \right) \tag{5.22}$$

where $e_\gamma^m = \min \left\{ \gamma_L - \frac{13}{16}(\gamma_L)^2, \gamma_l - \frac{\alpha d_{l+1} w_m^l}{20} (\gamma_l)^2 \right\}$ *for* $l = 1, \ldots, L-1$

Remarks: The structure of the bound in (5.22) is, in principle, similar to those from (5.9) and (5.18). Hence, the trends suggested by the interplay of the hyper-parameters – the deviation D_f, the network depth L and sizes $(d_0, \ldots, d_L)$ including stepsizes, N and B for learning L-NN are clearly similar to those observed from the Theorem 5.1 and Corollary 5.8 (from (5.9) and (5.18) respectively). Specifically, the discussions about denoising rates and data statistics, and their role in parameter convergence will still apply here for the layer-wise pretraining stage. It is interesting to see that the output layer influence (the

second term in (5.22)) needs to be completely compensated by the batchsize. Note that this bias is, in principle, similar to the terms in (5.9) and (5.18). Also (5.22) uses the same (α, δ_α) for all layers i.e., all the $L-1$ hidden layers are pretrained to the same (α, δ_α) solution. Instead, one may use different α_ls for each of these layers. The modified decay of gradients bound, which follows from simple adjustments in the steps involved in constructing (5.22) is,

$$\mathbb{E}_R\|\nabla_{\mathbf{W}}\tilde{f}(\mathbf{W}^R)\|^2 \leqslant \frac{1}{e_\gamma^m}\left(\frac{D_f}{N}+\frac{1}{B}(e_L^m+\sum_{l=1}^{L-1}\alpha_l e_l^m)\right)$$
$$\text{with} \qquad e_\gamma^m = \min\left\{\gamma_L - \frac{13}{16}(\gamma_L)^2, \gamma_l - \frac{\alpha_l d_{l+1} w_m^l}{20}(\gamma_l)^2\right\} \tag{5.23}$$

This setting is of interest when comparing two networks (in Section 5.5) and later in Section 5.8 where we discuss the design choice problem. Because of the presence of composition of multiple layers, the interplay of network structure and decay of expected projected gradients is much more involved than for 1-NN or DA. The easiest way to interpret them is, clearly, by visualizing the trends, as will be presented in Section 5.7, but following the presentation in Section 5.4, we discuss some typical L-NN settings of interest before moving on to dropout.

Goodness of Pretraining

We see from Definition 5.5 and Theorem 5.11 that α and δ_α control the goodness of layer-wise pretraining. Hence, the influence of the $L-1$ DAs, and the relevant hyper-parameters (from Theorem 5.8), are subsumed within α, δ_α and D_f. For fixed network lengths and stepsizes, e_l^m are constants and encode the variance of gradients within the l^{th} layer and the corresponding free parameters. As implied by the terms in (5.22), $\alpha \approx 0$ will ensure that the backpropagation is not influenced by the structure of these $L-1$ pretrained layers. With such a small α, consider two different learning regimes depending on the deviation of the pretrained network D_f.

1. **Small D_f:** If the deviation is small, then the supervised tuning is mostly confined to mapping $\mathbf{h}^{L-1}$ to the outputs $\mathbf{y}$. This is the *ideal* multi-layer NN setting where the (learned) higher level representations are already very good (even before tuning them to the corresponding $\mathbf{y}$s) in predicting the output labels. Here, (5.22) predicts that both B and N may be reasonably small, and under the assumption that $SC \approx BN$ (which was used in the Remarks of Theorem 5.6), this implies that a small number of labeled

instances are sufficient. However, forcibly restricting the backpropagation to only learning the last layer is not necessarily good, and one may instead want to allow for all $\mathbf{h}^l$s ($l = 1, \ldots, L-1$) to change if needed (Vincent et al., 2010; Saxe et al., 2011).

2. **Large D_f:** In the alternate setting with large D_f, although the pretrained $\mathbf{h}^l$s are good enough representations of the inputs, the desired outputs are not well modeled yet. In such a case, aggressive tuning is needed, as suggested by (5.22), with large N and B.

Overall, (5.22) implies that the goodness (or the lack thereof) of pretraining will be passed on to the tuning stage convergence via α and D_f. Clearly, as network size increases, e_l^ms increase proportionally, thereby requiring large N and batchsizes B. Recently, (Yosinski et al., 2014) showed empirical evidence that aggressive pretraining (i.e., $\alpha \approx 0$), especially in higher layers, results in $\mathbf{h}^l$s that cannot be translated across multiple arbitrary tasks. Theorem 5.11 suggests the same – to allow for aggressive tuning instead of aggressive pre-training, α and D_f should not be small. So, if the goal is to generate representations that are *general enough* to be applicable to a wide-variety of related, but reasonably distinct tasks, then it makes sense to perform weak pretraining with large $\alpha \not\approx 0$, while increasing the supervised gradient iterations N.

Comparing arbitrary networks

(Yosinski et al., 2014) and others have investigated empirically whether the lower level representations or the higher ones are better translatable across tasks. From the convergence perspective, Theorem 5.11 provides a clear answer to this, including other open questions listed in Section 5.1, for instance, whether two given deep architectures are *equivalent* and when can one truncate the hidden layers without compromising the expected gradients (and eventual parameter estimation). Very recently, (Wei et al., 2016) addressed such equivalence and morphism properties of deep networks. To pursue these questions, consider a more general setting where the $L-1$ hidden layers are trained to α_l ($l = 1, \ldots, L-1$) different goodness levels. Such a setting with different α_ls is only interesting when comparing two networks, and therefore, was not used for Theorem 5.11 and the discussion in Section 5.5. Without loss of generality, let $\gamma^l = \gamma$, $w_m^l = w \forall l$ and $D_f \approx d_L$ (since $\mathbf{y} \in [0,1]^{d_L}$). With this setup, a deep network $\mathcal{D}$ is parameterized by the remaining hyper-parameters (α_l, d_l)s, B, N and the stopping distribution $\mathcal{P}_R(\cdot)$.

Consider learning two different deep networks $\mathcal{D}_1$ and $\mathcal{D}_2$ for datasets sampled from an underlying data generating distribution i.e., for a fixed d_0 and the data statistics (μ_x and τ_x). Clearly, when comparing different networks that learn the same concept, it is reasonable to estimate the difference in objectives by comparing their generalization errors. However, in addition, we look at the parameter estimation (and the resulting convergence) aspect as well. So, we instead use the bound in (5.22) to decide whether $\mathcal{D}_1$ and $\mathcal{D}_2$ are *close enough* in terms of learning the underlying concept. While presenting the design principles (Section 5.8) and resulting evaluations on designing networks for a given task we discuss comparing the learning objectives and generalization error. We have the following definition.

Definition 5.12 (**Parameter convergence of two deep networks**). *Two arbitrarily different networks $\mathcal{D}_1$ and $\mathcal{D}_2$ learned on two datasets sampled from some data generating distribution are equivalent in terms of parameter convergence, denoted by $\mathcal{D}_1 \equiv_e \mathcal{D}_2$, whenever the bound in* (5.22) *is the same for both.*

Our discussion of which families of networks offer no better convergence guarantees over the other will be driven by the above definition. It should nevertheless be noted that, whenever (5.22) is loose, such insights might not hold – we discuss these issues in Section 5.8.

Different networks of same depth

There are many different settings for α_ls, d_ls and other hyper-parameters for which two equal depth networks will be equivalent (according to Definition 5.12). The following result summarizes two such general families of networks that guarantee the same decay of expected gradients, pointing out a few special cases of interest. To restrict the comparison to pretraining and structural aspects, we fix B, N and $\mathcal{P}_R(\cdot)$ across different networks (including the nature/type of activation functions used). For this result, we fix the output layer and the hidden layer that feeds into it (d_{L-1} and d_L), which will be relaxed later when comparing dropout networks in Section 5.6.

Corollary 5.13 (**Equivalence of networks**). *Consider the family of depth L deep networks that learn datasets from an underlying distribution (i.e., given d_0, μ_x and τ_x) using multi-layer RSG with no dropout and a given B, N, γ_l, w_m^l and stopping iteration R. All such networks $\mathcal{D} := (\alpha_l, d_l)$, with $\alpha_l d_{l+1} < \frac{65}{4}$ for $l = 1, \ldots, L-1$ and*

$$\alpha_l d_{l-1} d_l d_{l+1} = \frac{1}{\Psi_l} \quad \textit{for a given } \Psi_1, \ldots, \Psi_{L-1},\ d_L \textit{ and } d_{L-1} \tag{5.24}$$
are equivalent to each other according to Definition 5.12.

Remarks: We first note that the notion of equivalence here is "only" based on parameter convergence (i.e., the decay of expected gradients) as described in Definition 5.12. We do not make any claim about its relationship with statistical equivalence (and other variants). The terms on the left hand side in (5.24) come from the structural hyper-parameters e_l^m (see Theorem 5.11). Hence, the constants $\Psi_1, \ldots, \Psi_{L-1}$ correspond, in some sense, to the contribution of each of 1-layered networks that compose a L-NN to the eventual parameter convergence. Any network that belongs to the family of Corollary 5.13 i.e., any choice of α_ls and hidden layer lengths d_ls that satisfy (5.24), for a given B, N, d_0 and d_L will have the following decay of expected gradients (let $\gamma_l = \gamma$ and $w_m^l\ w \forall l$),

$$\mathbb{E}_R \|\nabla_{\mathbf{W}} \tilde{f}(\mathbf{W}^R)\|^2 \leqslant \frac{1}{e_\gamma^m} \left(\frac{D_f}{N} + \frac{1}{B} \left(e_L^m + \frac{\gamma w}{4} \sum_{l=1}^{L-1} \frac{1}{\Psi_l} \right) \right) \tag{5.25}$$

where both $e_\gamma^m = \gamma - \frac{13}{16}\gamma^2$ and $e_L^m = \frac{13 d_{L-1} d_L \gamma^2}{256}$ are constants for all networks of this family. Thereby, $\Psi_1, \ldots, \Psi_{L-1}$ can be interpreted as the *tuning goodness criteria*, and (5.24) can be used to design the network i.e., *choose* hidden layer lengths to attain a given convergence level. Specifically, given Ψ_ls and the pretraining goodness criteria α_ls, (5.24) can be used to recursively compute d_ls. Such strategies are discussed in detail in Section 5.8 where (5.24), along with its dropout counterpart from Section 5.6, provide a systematic way to address the design choice problem from Section 5.1. Observe that d_{L-1} and d_L were fixed when discussing the equivalence of networks in the above result. This is because Corollary 5.13 is driven by the layer-wise pretraining regime, where, the influence of $L-1$ pretrained layers is passed on to the tuning stage via the $L-1$ constraints in (5.24). This restriction is relaxed when discussing equivalence of dropout networks later in Section 5.6. We note that, beyond Corollary 5.13, there are several alternate settings of (α_l, d_l)s that lead to equivalent networks but do not belong to the family characterized by the above result.

Here, we list out one interesting network design 'modulations' that one applies while seeking to identify the best deep network for a given problem, which provide the same level of convergence, independent of the generalization.

Let $\mathcal{D}_1 := (\alpha_{l,1}, d_{l,1})$ be the original network, and $\mathcal{D}_2 := (\alpha_{l,2}, d_{l,2})$ be the modulated one.

Scaling $d_{l,1}$s and $\alpha_{l,1}$s : For any $\varphi > 0$, $\mathcal{D}_1 \equiv_e \mathcal{D}_2$

$$d_{l,2} = \varphi d_{l,1} \quad \text{for} \quad l = 1, \ldots, L-2 \quad \text{and} \quad d_{L-1,2} = d_{L-1,1}$$
$$\alpha_{1,2} = \frac{\alpha_{1,2}}{\varphi^2} \quad \alpha_{l,2} = \frac{\alpha_{l,1}}{\varphi^3} \quad \text{for} \quad l = 2, \ldots, L-2 \quad \text{and} \quad \alpha_{L-1,2} = \frac{\alpha_{L-1,1}}{\varphi} \tag{5.26}$$

These choices of (α_l, d_l)s follow from Corollary 5.13. Recall that there is strong evidence for choosing large hidden layers (Bengio, 2009a; Hinton, 2010; Bengio, 2012). The identity in (5.26) suggests that depending on how the pre-training criteria are changed, the newer network with changed hidden layer lengths might not necessarily improve the parameter convergence. Alternatively, using (5.26) (and its parent (5.24)) the choices of α_l and φ that do not guarantee improved convergence may be avoided if needed. It is not reasonable to use very large or very small φ (networks with too small hidden layers are unnatural and very restrictive), and the equivalence suggested by (5.26) is non-informative for such unreasonable (and uncommon) networks. An interesting observation from (5.26) is that any change in small set of layers in the network can, in principle, be compensated with appropriate changes in the other layers. In other words, arbitrarily scaling up the layer lengths is not beneficial. A more closer discussion about this observation follows shortly. Some of these trends inferred from (5.26) seem intuitive, and the above set of results (Corollary 5.13, (5.25) and (5.26)) are, to our knowledge, the first to summarize such observations.

Corollary 5.13 fixes the hidden layer that feeds into the output layer (i.e., d_{L-1} is fixed). Building upon the observations made earlier, i.e., the convergence (expected gradients) gains because of changes in some layers may, inadvertently, be compensated by changes in the rest, the following result removes this restriction on the $L-1^{th}$ layer. It presents the interesting scenario where one layer can, in principle, be a bottleneck for the entire network. (Yosinski et al., 2014) among others analyze the importance of lower layers and their learnability in producing representations that are generalizable. To that end, we consider the setting where (α_l, d_l)s for the two networks $\mathcal{D}_1$ and $\mathcal{D}_2$ are chosen arbitrarily such that $\alpha_{l,1} d_{l+1,1} = \rho_1 < \frac{65}{4}$ and $\alpha_{l,2} d_{l+1,2} = \rho_2 < \frac{65}{4}$ for $l = 1, \ldots, L-2$ (for some given ρ_1 and ρ_2). Comparing the resulting (5.22) for $\mathcal{D}_1$ and $\mathcal{D}_2$, we get the following

$$
\begin{aligned}
\mathrm{d}_{\mathrm{L}-1,2} &= \frac{1}{\Upsilon_2}(\Upsilon_1 \mathrm{d}_{\mathrm{L}-1,1} + \Upsilon_3) \\
\Upsilon_1 = \frac{13}{256}\mathrm{d}_{\mathrm{L}} + \frac{\rho_1}{4}\mathrm{d}_{\mathrm{L}-2,1} \quad \Upsilon_2 &= \frac{13}{256}\mathrm{d}_{\mathrm{L}} + \frac{\rho_2}{4}\mathrm{d}_{\mathrm{L}-2,2} \\
\Upsilon_3 &= \sum_{\mathrm{l}=1}^{\mathrm{L}-2} (\rho_1 4 \mathrm{d}_{\mathrm{l}-1,1}\mathrm{d}_{\mathrm{l},1} - \rho_2 4 \mathrm{d}_{\mathrm{l}-1,2}\mathrm{d}_{\mathrm{l},2})
\end{aligned}
\tag{5.27}
$$

The above set of equalities follow by rearranging terms in (5.22). Although (5.27) shows the relationship between the two $\mathrm{L}-1^{\mathrm{th}}$ hidden layers of $\mathcal{D}_1$ and $\mathcal{D}_2$, a similar set of equalities can be obtained for any other hidden layer. First, (5.27) implies that, for a given B, N and dataset, the convergence can be very sensitive to the choices of hidden layer lengths. The strong dependence of (5.9), (5.16), (5.18) and (5.22) on the layer lengths does provide some evidence for this sensitivity, but the above set of equalities summarize it in much finer detail. Second, (5.27) says that two arbitrary equal depth networks with possibly different α_{l}s and hidden layer lengths, can be 'forced' to be equivalent to each other in terms of parameter convergence by simply choosing *one* of the layers according to (5.27). Beyond the remarks for Corollary 5.13, this is additional evidence that arbitrary (reasonably large) choices of hidden layers are not necessarily good.

Further, (5.27) predicts several of the observations made earlier in (Yosinski et al., 2014; Long et al., 2015), where the authors present empirical evidence explaining the influence of lower versus higher layers on generalization. In particular, observe that (5.27) simply says that any gain in parameter convergence by changing lower layers can, in principle, be undone by changing the higher layers – in the worst case, one layer can act as the bottleneck. Now, because of the structure of the loss function, the backpropagation algorithm moves the lower layer (layer closer to inputs) estimates slower than higher ones (layer closer to outputs). Hence, the lower layers, from the view point of parameter convergence, are more translatable across tasks. The specific rate of this translatability may be computed using the constants Υ_1, Υ_2 and Υ_3 – the higher they are, the more sensitive they are to predicting $\mathbf{y}$s, and hence less translatable. The argument for which layers regularize the network better may not be directly answerable by this discussion. The authors of (Erhan et al., 2010b) argue that pretraining is an unusual form of regularizer, while being an efficient warm-start for backpropagation. Definition 5.12, Corollary 5.13 and the resulting discussion in this section categorizes some such families

of pretrained networks that provide the same warm-start behavior, which as one may expect, should facilitate addressing the deep network design choice problem (see Section 5.8).

Fatter vs. Thinner vs. Taller vs. Shorter Networks:

The observations made earlier, in Section 5.4, about thinner networks being preferable to fatter ones can be seen from e_l^ms in (5.22), which clearly suggests that the network depth should *not* be increased beyond necessary – similar to Occam's Razor. Such a minimum depth will depend on the trade-off between convergence, here in the form of expected gradients from (5.22), and some performance measure like generalization (Hardt et al., 2015). To explore these insights more closely, we now consider the case where the two networks under comparison are of different depths, and loosely define four categories of L-NNs:

Fat network: Large d_l, Large L *Thin network:* Small d_l, Small L

Tall network: Small d_l, Large L *Short network:* Large d_l, Small L

Here l corresponds to the hidden layers (input and output layer lengths are given). Clearly, this is not a formal definition – the terms *large* and *small* are not absolute, and will be used in the context of comparing two networks. A simple observation from (5.24) and (5.25) is that, for a given dataset, N, B and other learning parameters, fatter networks have a much slower decay of gradients compared to others. To compare a tall network $\mathcal{D}_t$ to a short one $\mathcal{D}_s$ (with $L_s < L_t$ and larger hidden layer lengths) consider the setting where all hidden layers are of the same length, denoted by d_t and d_s respectively (with $d_s > d_t$). With this definition we have the following result for their equivalence,

Corollary 5.14 (Taller vs. Shorter networks). *Given a short network* $\mathcal{D}_s := (\alpha, d_s)$ *of depth* L_s. *Consider a tall network* $\mathcal{D}_t := (\alpha, \frac{d_s}{1+\delta_{st}})$ *of depth* $L_t > L_s \geqslant 3$ *with* $\delta_{st} > 1$. *Both networks learn datasets from the same distribution with a given* d_0, d_L, μ_x *and* τ_x, *using multi-layer RSG with the same* B, N, γ, w *and stopping iteration* R. *Let* $\alpha d_s < \frac{65}{4}(1+\delta_{st})$. $\mathcal{D}_t$ *is better than* $\mathcal{D}_s$ *in terms of* $\equiv_e$ *from Definition 5.12, only if*

$$L_s < L_t \leqslant L_s + 3\delta_{st}(L_s - 3) + \Delta \tag{5.28}$$

where $\Delta = \delta_{st}(1+\delta_{st})^2 \left(\frac{e_1^m + e_{L-1}^m}{e_2^m} + \frac{e_L^m}{\alpha e_2^m} \right)$, *and* e_1^m, e_2^m, e_{L-1}^m *and* e_L^m *are the structural constants of* $\mathcal{D}_s$.

Remarks: This is an interesting result which says that in certain cases taller networks have faster convergence than shorter ones. Stacking up large number of layers is one of the extensively used design criteria to improve the generalization performance of deep networks. First, (5.28) implies that, whenever the depth of the larger network $\mathcal{D}_t$ is within a certain limit (and when the hidden layers of the $\mathcal{D}_t$ are smaller than $\mathcal{D}_s$), the taller network has a faster decay of gradients. This leads to a faster training time (smaller N) for $\mathcal{D}_t$, which using the result from (Hardt et al., 2015), implies that generalization improves. From the perspective of parameter convergence, this is the first clear justification for the necessity of adding more layers. Second, (5.28) says that the gains in performance can only be retained if the depth is below a certain upper limit. This is an interesting insight, with practical implications when applying deep networks on new tasks or changing the existing network for better performance. For instance, consider a depth $\mathsf{L}_s = 5$ network $\mathcal{D}_s := (0.01, 100)$ learned on a dataset with $\mathsf{d}_0 = 100$, $\mathsf{d}_{\mathsf{L}} = 10$ (let $\gamma = 0.5$ and $w = 0.1$). A new network $\mathcal{D}_t$ with say $\mathsf{d}_t = \frac{2\mathsf{d}_s}{3}$) will be better in terms of parameter convergence only if its depth $\mathsf{L}_t \lesssim 9$.

Apart from the results in the previous sections, it is clear from (5.22), that we can control the *convergence* of multi-layer nets (in terms of parameter estimation) by preceding the backpropagation tuning with layer-wise pretraining. There is already strong empirical evidence for the generalization performance of pretraining (Erhan et al., 2010b; Vincent et al., 2010; Saxe et al., 2011; Bengio et al., 2007), and Theorem 5.11 complements these studies with guaranteed convergence of gradients. The trends and trade-offs predicted in Sections 5.5 and 5.5 can be related to the bounds on training times (derived from generalization (Hardt et al., 2015)). In Sections 5.7 and 5.7 we show more evidence for these bounds, deriving more subtle trends which are, for the most part, not observable in broad empirical studies by simulating the bounds. We now adapt the result in Theorem 5.11 to the most general L-NN learning setting, leading to new and interesting observations about multi-layer deep networks. The landscape of deep learning literature moves swiftly, and since the proposal of our ideas there have been several works on local and global optimality in deep networks. We refer the reader to Swirszcz et al. (2017); Du et al. (2017); Haeffele and Vidal (2017); Keskar et al. (2016).

5.6 Multi-layer with Dropout

Using ReLUs and dropout, (Nair and Hinton, 2010; Krizhevsky et al., 2012; Szegedy et al., 2014) and others have shown that whenever large amounts of labeled data are available, pretraining might not be *necessary*, and can be completely by-passed, to achieve good generalization, This can be partly seen from (5.22) in Theorem 5.11 (and the corresponding remarks). The experiments reported in these studies clearly suggests a potential relationship between dropout and layer-wise pretraining. To explore this interplay, the following result summarizes the convergence of expected projected gradients in the most general setting. Here, the hidden layers are first pretrained (as in Section 5.5), and the pretrained network is then tuned via supervised backpropagation *in tandem* with dropout induced in the input and hidden layers. Observe that dropout is only used during this tuning phase. A special case of this general setting will be fully-supervised L-NN dropout learning with *no pretraining* and random initializations for $\{\mathbf{W}^1, \ldots, \mathbf{W}^L\}$ from (Srivastava et al., 2014). As mentioned while introducing the loss in (5.6), $\zeta_0, \ldots, \zeta_{L-1}$ denotes the dropout (i.e., w.p. $1 - \zeta_l$ the activation is 0: it is dropped) of the input and the $L-2$ hidden layers. Since α controls the goodness of pretraining (see Theorem 5.11), it is reasonable to expect some interaction between α and ζ.

Theorem 5.15 (Pretraining vs. Dropout). *Given the dropout rates $\zeta_0, \ldots, \zeta_{L-1}$, for learning the L-layered network from Theorem 5.11 which is pretrained to a (α, δ_α) solution, we have*

$$\mathbb{E}_R \|\nabla_{\mathbf{W}} \tilde{f}(\mathbf{W}^R)\|^2 \leqslant \left(\frac{D_f}{N e_\gamma^m \underline{\zeta}^2} + \frac{1}{e_\gamma^m B} \left(\frac{\zeta_{\mathcal{B}} e_L^m}{\underline{\zeta}} + \alpha \zeta_{\mathcal{B}}^2 e_{L-1}^m + \alpha \zeta_{\mathcal{B}}^2 \sum_{l=1}^{L-2} e_l^m \right) \right) \tag{5.29}$$

where $\underline{\zeta} = \min_l \zeta_l$ and $\zeta_{\mathcal{B}} = \frac{\max_l \zeta_l}{\min_l \zeta_l}$ for $l = 0, \ldots, L-1$

Remarks: This is our most general result. Although, the concepts of layer-wise pretraining and dropout have been studied independently, to our knowledge, this is a useful (and possibly the first) result that explicitly combines these two regimes. Specifically, (5.29) relates these regimes in a systematic way, and their *joint* influence on the convergence of the gradients, and to the 'achievable' generalization. Recall that $e_1^m, \ldots, e_L^m$ encode the degrees of freedom of the L-NN. Hence, the first term in (5.29) corresponds to the goodness-of-fit of the

outputs, while the other terms represent the effective 'reduction' in the number of free parameters because of pretraining.

An important hyper-parameter in Theorem 5.15 is $\zeta_{\mathcal{B}}$, the ratio of the maximum and minimum dropout rates. Clearly, $\zeta_{\mathcal{B}} \geqslant 1$, and as the variance of ζ_ls increases, $\zeta_{\mathcal{B}}$ increases accordingly, requiring larger batches and iterations. We refer to $\zeta_{\mathcal{B}}$ as the *dropout bottleneck ratio*, or simply the bottleneck ratio, since it corresponds to the variability in the fraction of unknowns (i.e., the fraction of dropped units) being updated across the L layers. Note that this is unrelated to the information bottleneck in multi-layer neural networks (Tishby and Zaslavsky, 2015). The last three terms of (5.29) show that, $\zeta_{\mathcal{B}}$, $\underline{\zeta}$ (the smallest dropout rate across all layers) and the pretraining goodness α are critical in reducing this number of free parameters (and consequently, the induced bias from the stochastic gradients) after pretraining. For a fixed dataset (and B, N), the bottleneck ratio can be used as a surrogate to how 'bad' the estimates $\mathbf{W}$ may be, or more specifically, how much variability there might be across different runs of multi-layer RSG. In the special case where $\zeta_l = \zeta$ for $l = 0, \ldots, L-1$, (5.29) reduces to

$$\mathbb{E}_R \|\nabla_{\mathbf{W}} \tilde{f}(\mathbf{W}^R)\|^2 \leqslant \left(\frac{D_f}{N e_\gamma^m \zeta^2} + \frac{1}{e_\gamma^m B} \left(\frac{e_L^m}{\zeta} + \alpha e_{L-1}^m + \alpha \sum_{l=1}^{L-2} e_l^m \right) \right) \tag{5.30}$$

but as we show later in Corollary 5.16, a tighter bound can be derived for this constant dropout setting. Note that (5.29) is not the most general form of the decay bound for pretraining versus dropout. Instead of using $\underline{\zeta}$ and $\zeta_{\mathcal{B}}$, as was done in Theorem 5.15, one can construct the bound by directly using the dropout rates $\zeta_0, \ldots, \zeta_L$. Such a bound is harder to interpret and involves an influence from all the different hyper-parameters, and has little to offer over (5.29). To better interpret (5.29), while deriving interesting trends and inferences about pretraining versus dropout in the context of convergence, we further simplify Theorem 5.15 by using the same dropout rate across all the layers i.e., $\zeta_l = \zeta$ for $l = 0, \ldots, L-1$. The following result summarizes the new bound. This is tighter than simply imposing $\zeta_l = \zeta$ in (5.29), because the relaxation in the proof of Theorem 5.15 to make the result more interpretable can now be removed.

Corollary 5.16 (Pretraining vs. Dropout ($\zeta_l = \zeta$)). *Whenever the input and hidden layers have the same dropout rate $\zeta_l = \zeta$ for $l = 0, \ldots, L-1$, for learning the L-layered network from Theorem 5.11 which is pretrained to a (α, δ_α) solution, we have*

$$\mathbb{E}_R \|\nabla_{\mathbf{W}} \tilde{f}(\mathbf{W}^R)\|^2 \leqslant \frac{1}{e_\gamma^m} \left(\frac{D_f}{N\zeta^2} + \frac{1}{B} \left(\frac{e_L^m}{\zeta} + \alpha e_{L-1}^m + \alpha\zeta \sum_{l=1}^{L-2} e_l^m \right) \right) \tag{5.31}$$

We see that (5.31) has the same structure as (5.29) but is simpler. Given the network structure (L and $d_0, \ldots, d_L$) and stepsizes ($\gamma^1, \ldots, \gamma^L$), the hyper-parameter functions $e_1^m, \ldots, e_L^m$ are constants (see Theorem 5.11). D_f does depend on the goodness of pretraining, but will nevertheless be large because the loss is calculated over the extra output layer between the predicted and the observed **y**s (see (5.6)). With the above result, consider the two standard learning mechanisms — dropout learning with and without layer-wise pretraining.

Pretraining + Dropout:

1. **Small α:** If the network is pretrained to a reasonably small α, the last two terms in (5.31) are already as small as they can be, and will be dominated by the first two terms. In this setting, for a given B and N, the best choice for ζ will then be ≈ 1 i.e., very minimal or dropout. It is known that this procedure works very well — this is essentially "good" layer-wise pretraining followed by supervised fine-tuning (Hinton and Salakhutdinov, 2006; Erhan et al., 2010b). The bound in (5.31) presented a clear theoretical argument for the initial use of pretraining in deep networks from the perspective of convergence and eventually, generalization.
2. **Large α:** Alternatively, if the pretraining is poor (i.e., large α), and we still operate in the $\zeta \to 1$ regime, the fine-tuning process will update the full network in each iteration and may result in overfitting. This is the fundamental argument that motivated the work on dropout learning (Srivastava et al., 2014). This argument was made empirically earlier in this work and others (Baldi and Sadowski, 2014), and our result in Corollary 5.16 provides a complete justification for it. The cost of using dropout though is slower convergence because the first and second terms will rapidly increase depending on ζ. This is expected since dropout essentially adds 'noise' to the solution path, forcing backpropagation to work with a subset of all activations (Wager et al., 2013), and overall, this involves more work. Dropout dynamics studied in (Rifai et al., 2011; Baldi and Sadowski, 2014) also make this observation but *not* from the perspective of parameter convergence.

Independent of the precise dropout rate ζ, (5.31) clearly implies that pretraining always improves convergence (since α will reduce). (Abdel-Hamid et al., 2013; Lee et al., 2009; Srivastava et al., 2014) have shown empirically that this is

especially true for improving generalization. The optimal choices of ζ are discussed later in this Section 5.6.

Only Dropout: When pretraining is not allowed, there is a complicated trade-off between the terms involving ζ in (5.31). Specifically, the optimal ζ will depend on balancing the variance from the hidden layers (the last term) versus the goodness of approximation of the outputs (first and second terms). For certain values of D_f and e_l^m's, the best ζ will be ~ 0.5, which was reported as the best rate as per dropout dynamics (Baldi and Sadowski, 2014). The visualizations in Section 5.7 will provide more insight into this interplay. Large values of B and N will ensure that the bound is small, and a large N will, in turn, demand a large training dataset size (refer to the setup from Theorems 5.6, 5.8 and 5.11). Put another way, if large amounts of labeled data are available, one can by-pass pretraining completely and only perform supervised backpropagation forcing all the terms in (5.31) to reduce with reasonably large B and number of epochs. This was the argument put forth by recent results in deep learning (Krizhevsky et al., 2012; Szegedy et al., 2014; Ciregan et al., 2012) — performing fully supervised dropout with very large amounts of training data where pretraining has been for the most part discarded completely.

Computational Complexity of multi-layer RSG

Theorems 5.6 and 5.10 described the iteration complexity, i.e., the number of maximum allowable gradient iterations N, for single-layer and DA RSG respectively. As observed in Theorem 5.15 and Corollary 5.16, the convergence bound for multi-layer dropout is complicated involving various learning and network hyper-parameters. Nevertheless, under the reasonable assumption that $SC \approx BN$ where S is the sample size (training dataset size) and C denotes the number of epochs, the following result calculates the minimum number of training instances required to ensure a (ϵ, δ) solution (see Definition 5.5) for multi-layer RSG with dropout. Note that the same assumption was used in the Remarks of Theorem 5.6 to discuss some example sample sizes.

Theorem 5.17 (**Multi-layer RSG Computational Complexity**). *Consider a multi-layer RSG from Theorem 5.11 with input and hidden layer dropout rate of ζ and the network is pretrained to a (α, δ_α) solution. To obtain a (ϵ, δ)-solution, we need*

$$B \geqslant \frac{2\Pi}{\epsilon_\gamma^m \epsilon \delta^{1/T}} \quad \textit{and for such a choice of batchsize} \quad N \geqslant \max\left\{\frac{SC}{B}, \frac{2d_L}{\epsilon_\gamma^m \zeta^2 \epsilon \delta^{1/T}}\right\} \tag{5.32}$$

where C *is the number of backpropagation epochs,* S *is the number of labeled training instances and* $\Pi = \left(\frac{e_L^m}{\zeta} + \alpha e_{L-1}^m + \alpha\zeta \sum_{l=1}^{L-2} e_l^m\right)$

Remarks: As mentioned in the remarks of Theorem 5.10, larger networks entail large batchsizes. Nevertheless, if the networks is pretrained to small α this requirement can be relaxed (see (5.32)). Similar to (5.21), D_f has been replaced by D_L there by making the bounds in (5.32) deployable in practice. The design procedures in Section 5.8 will use this for computing the required B and N.

Choosing the dropout rates

The authors in (Baldi and Sadowski, 2014) exhaustively analyzed the learning dynamics of the dropout algorithm. They suggest that, independent of the network lengths or size, $\zeta = 0.5$ is optimal in the learning theoretic sense i.e., the resulting dropped representations are unbiased and asymptotically equivalent to the ideal 'noise-injected' representations. Our results in (5.29) and (5.31) suggest that choosing 0.5 dropout is good but *may not* always be ideal to guarantee the best decay of expected gradients. Such an optimal ζ clearly depends on the network structure and data statistics. To analyze (5.31) (which assumes $\zeta_l = \zeta$ for $l = 1, \ldots, L-1$) more closely, we use the running example from Section 5.5, with constant stepsizes $\gamma^l = \gamma$, a fixed box-constraint across all layers $w_m^l = w \forall l$ (for some $w > 0$) and assume $D_f \approx d_L$ (since $\mathbf{y} \in [0,1]^{d_L}$). The following result presents the best $0 < \zeta < 1$ for the network with same hidden layer lengths $d_l = d$ for $l = 1, \ldots, L-1$.

Corollary 5.18 (**Optimal dropout rate**). *The best constant dropout rate* $\zeta = \zeta_l \ \forall l$ *for learning a* L*-NN with a constant hidden layer length* d*, using multi-layer RSG from Algorithm 4 with* $\gamma^l = \gamma$, $w_m^l = w > 0 \forall l$ *and* $\lceil \frac{1}{\kappa} \rceil$ *number of epochs, is given by*

$$\textit{median}\left\{0, \sqrt[3]{\frac{C_1}{C_2}}, 1\right\} \quad \textit{where} \quad C_1 = \left(d_L + \frac{13d^2\gamma^2}{256\kappa}\right) \quad \textit{and} \quad C_2 = \frac{\alpha\gamma w}{4\kappa}\left(d_0 d^2 + (L-3)d^3 + d^2 d_L\right) \tag{5.33}$$

Remarks: We visualize these optimal rates in Section 5.7 for several choices of the hyper-parameters involved in C_1 and C_2. Clearly, ζs which are too small or too large may not be reasonable from the perspective of dropout dynamics (Baldi and Sadowski, 2014). The interaction of α and ζ, discussed above in (5.33) formalizes several such observations. A large α clearly pushes the best ζ to be closer to zero, i.e., the network will need to use large dropout rates when either (1) the pretraining is not effective or (2) the layers have not been pretrained to begin with. Consider a network with a reasonably large d (compared to d_L) and number of epochs $\lceil \frac{1}{\kappa} \rceil$. In this setting, the constants C_1 and C_2 from (5.33) can be approximated as

$$C_1 = \frac{13 d^2 \gamma^2}{256 \kappa} \quad , \quad C_2 = \frac{\alpha \gamma w}{4\kappa}(L-3)d^3 \quad \text{and the best} \quad \zeta \approx \sqrt[3]{\frac{0.2\gamma}{\alpha w (L-3) d}} \tag{5.34}$$

For instance, for the example network $\mathcal{D}_s$ used in the remarks of Corollary 5.14 with $d = 100$, $\gamma = 0.5$, $w = 0.1$ and $\alpha = 0.01$, using (5.34), the optimal ζ is approximately 0.79 i.e., in each iteration of the backpropagation, only 20% of all units can be nullified and the updates can be restricted to the remaining 80% of the whole network. When pretraining is poor (or not utilized) and α increases to, say 2, from 0.01 (the latter is the well pretrained network), the optimal ζ decreases to 0.13 and updates must be restricted to a much smaller part of the whole network.

Comparing arbitrary networks – with dropout

In Section 5.5, we used Theorem 5.11 to analyze the equivalence (see Definition 5.12) of arbitrary layer-wise pretrained multi-layer networks. Based on the discussion, equivalence results of equal length networks (Corollary 5.13) and arbitrary networks (Corollary 5.14) were obtained. We replicate this set of results for dropout networks using Theorem 5.15 and Corollary 5.16. Such results from Theorem 5.15, because of the presence of unequal dropout rates ζ_ls, lead to more complicated trends, and the resultant interpretation may not necessarily provide any new additional insight towards addressing the central question of comparing and designing deep networks on data generated from a given distribution. Hence, we use the constant dropout setting from Corollary 5.16. Similar to the setup from Section 5.5, consider learning two different deep networks for datasets sampled from some distribution, i.e., for a fixed d_0 and

the data statistics (μ_x and τ_x). We stick to the same general setting from Section 5.5 where the $L-1$ hidden layers have different pretraining goodness criteria α_ls. Following Definition 5.12, for dropout L-NNs $\mathcal{D}_1$ and $\mathcal{D}_2$ are equivalent whenever the bound in (5.31) is the same for either of them. Within this setting, we have the following result that characterizes two general families of networks that are equivalent.

Different networks of same depth – with dropout

Corollary 5.19 (**Equivalence of dropout networks**). *Consider the family of depth* L *deep networks that learn datasets from an underlying distribution (i.e., given* d_0, μ_x *and* τ_x*) using multi-layer RSG with dropout* ζ *and a given* B, N, γ_l, w_m^l *and stopping iteration* R. *All such networks* $\mathcal{D} := (\zeta, \alpha_l, d_l)$, *with* $\alpha_l d_{l+1} < \frac{65}{4}$ *for* $l = 1, \ldots, L-1$ *and*

$$\begin{aligned} &\zeta\alpha_l d_{l-1} d_l d_{l+1} = \frac{1}{\Phi_l} \qquad \alpha_{L-1} d_{L-2} d_{L-1} d_L = \frac{1}{\Phi_{L-1}} \\ &\frac{d_{L-1} d_L}{\zeta} = \frac{1}{\Phi_L} \qquad \frac{d_L}{\zeta^2} = \frac{1}{\Phi_f} \qquad \textit{for a given } \Phi_1, \ldots, \Phi_L \textit{ and } \Phi_f \end{aligned} \tag{5.35}$$

are equivalent ($\equiv_e$) *to each other according to Definition 5.12.*

Remarks: The setup of Corollary 5.19 is similar to Corollary (5.13). Similar to Ψ_ls from (5.24), Φ_ls and Φ_f denote the contribution of the structural hyper-parameters e_l^ms to the convergence bound – the tuning goodness criteria. Specifically, $\Phi_1, \ldots, \Phi_L$ correspond to the contribution of the L different 1-NNs that compose the multi-layer network, and Φ_f corresponds to the contribution from the initial deviation. Note that (5.35) has a total of $L+1$ such constraints. In the earlier non-dropout case from Corollary (5.13), Ψ_f and Ψ_L type terms were not used because d_l and d_{L-1} were fixed. The above result removes this restriction. Recall the remarks of Corollary 5.13 and (5.25) discuss how (5.24) can be used to design the network. Similarly, the more general form in (5.35) can also be used for designing dropout networks for a given d_0 and the pretraining and tuning criteria α_ls, Φ_ls and Φ_f. Any family of networks satisfying (5.35) will have the following decay of gradients

$$\mathbb{E}\|\nabla_{\mathbf{W}} \tilde{f}(\mathbf{W}^R)\|^2 \lesssim \frac{1}{e_\gamma^m} \left(\frac{1}{\Phi_f N} + \frac{1}{B} \left(\frac{13\gamma^2}{256\Phi_L} + \frac{\gamma w}{4} \sum_{l=1}^{L-1} \frac{1}{\Phi_l} \right) \right) \tag{5.36}$$

We defer the discussion on actually designing such families of networks to Section 5.8.

Fatter vs. Thinner vs. Taller vs. Shorter Networks:

The discussion in the previous section focused on networks of the same depth. Similar to Section 5.6 and Corollary 5.14, we have the following result for the taller versus shorter dropout networks, following the setup and discussion from Section 5.5. As observed with the non-dropout case in Section 5.6, fatter dropout networks have the slowest convergence.

Corollary 5.20 (Taller vs. Shorter dropout networks). *Given a short dropout network* $\mathcal{D}_s := (\zeta, \alpha, d_s)$ *of depth* L_s*. Consider a tall network* $\mathcal{D}_t := (\zeta, \alpha, \frac{d_s}{1+\delta_{st}})$ *of depth* $L_t > L_s \geqslant 3$ *with* $\delta_{st} > 1$*. Both networks learn datasets from the same distribution with a given* d_0*,* d_L*,* μ_x *and* τ_x*, using multi-layer RSG with the same* B*,* N*,* γ*,* w *and stopping iteration* R*. Let* $\alpha d_s < \frac{65}{4}(1+\delta_{st})$*.* $\mathcal{D}_t$ *is better than* $\mathcal{D}_s$ *in terms of* $\equiv_e$ *from Definition 5.12, only if*

$$L_s < L_t \leqslant L_s + 3\delta_{st}(L_s - 3) + \Delta \tag{5.37}$$

where $\Delta = \delta_{st}(1+\delta_{st})^2 \left(\frac{e_1^m}{e_2^m} + \frac{e_{L-1}^m}{\zeta e_2^m} + \frac{e_L^m}{\alpha\zeta^2 e_2^m}\right)$*, and* e_1^m*,* e_2^m*,* e_{L-1}^m *and* e_L^m *are the structural constants of* $\mathcal{D}_s$*.*

Remarks: The earlier statement about taller versus shorter networks (see remarks of Corollary 5.20) still applies here. In closing, we briefly summarize the main message. To our knowledge, (5.37) is potentially the first result justifying the increase of the network depth, in order to improve the convergence bound and consequently, the training time N, which, as pointed out for Corollary 5.20, in tandem with (Hardt et al., 2015), implies a better generalization. Consider the example network used earlier in the remarks of Corollary 5.14 but with a dropout rate $\zeta = 0.5$, i.e., $\mathcal{D}_s := (0.5, 0.01, 100)$ with $L_s = 5$ learned on a dataset with $d_0 = 100$, $d_L = 10$ (let $\gamma = 0.5$ and $w = 0.1$). A new dropout network $\mathcal{D}_t$ with say $d_t = \frac{2d_s}{3}$) will be better in terms of parameter convergence only if its depth $L_t \lesssim 10$.

Overall, the results in Sections 5.4 and 5.5 corroborate many existing observations about pretraining and dropout, and provides new tools to *guide* the learning procedure, in general. They provide insights into the convergence behavior, and allow us to *explicitly* calculate the best possible settings for all hyper-parameters. After evaluating the bounds in Section 5.7 and 5.7, we return to the design choice problem, motivated in Section 5.1, providing design *guidelines* for deep networks to guarantee certain level of parameter convergence and generalization on new learning tasks. Observe that the three algorithms pre-

sented in this work are only minor modifications over the classical autoencoder and backpropagation pipeline, and so, are very easy to implement.

5.7 Experiments

Apart from medical imaging data from ADNI dataset we will also utilize three widely studied computer vision dataset for evaluating the bounds suggested by (5.18), (5.22) and (5.31). This includes, MNIST: the hand written digits $(0-9)$ dataset with 70000 binary (black and white) images of size 28×28 (LeCun et al., 1998); CIFAR: object recognition dataset with 60000 color images, each of size 32×32 (Krizhevsky and Hinton, 2009); and CALTECH: another object categorization dataset with 5000 images of 32×32 pixels each (Fei-Fei et al., 2007). While ADNI data has two classes and 412 subjects each with MRI data (see Section 2.3 for details), the vision datasets correspond to multiclass classification. The datasets were rescaled appropriately for training networks with different input layer lengths (d_0). All features were normalized to $[0, 1]$. The evaluation trends were approximately the same for all four datasets, and so we summarize the results using a few representative plots. Figures 5.1 summarizes the behavior of expected gradients (or projected gradients), using different architectures and learning parameters. Figure 5.2 shows the influence of the stopping distribution, and Figure 5.3 shows the gap in the predicted versus observed trends. The y-axis (and colorbar, whichever appropriate) in the plots has been scaled by the corresponding maximum values to allow interpretation of the trends rather than the actual values. The batchsize $B = \min\{\lceil 0.1N \rceil, 100\}$ for all the experiments, and the expected gradients is simply the empirical average of gradients computed over the last $\min\{N, 50\}$ iterations (recall that N is the maximum number of allowable iterations). The data statistics (μ_x, τ_x) are given by $(0.13, 0.11)$ (MNIST), $(0.43, 0.30)$ (CIFAR), $(0.52, 0.35)$ (CALTECH) and $(0.14, 0.09)$ (ADNI).

Convergence versus d_x, d_h, w_m, ζ: Figure 5.1(a) shows the expected projected gradients of DA pretraining versus network sizes (hidden and input layer lengths). As predicted in (5.9) and (5.18), the convergence is slower for large networks, and visible and hidden layer lengths have unequal influence on the convergence (see remarks of Theorem 5.8). Here, $w_m = 1/\sqrt{d_v d_h}$ for all networks. The influence of denoising rate and w_m is shown in Figure 5.1(b) ($d_h = d_v$ for these networks). As suggested by (5.18), the convergence is faster

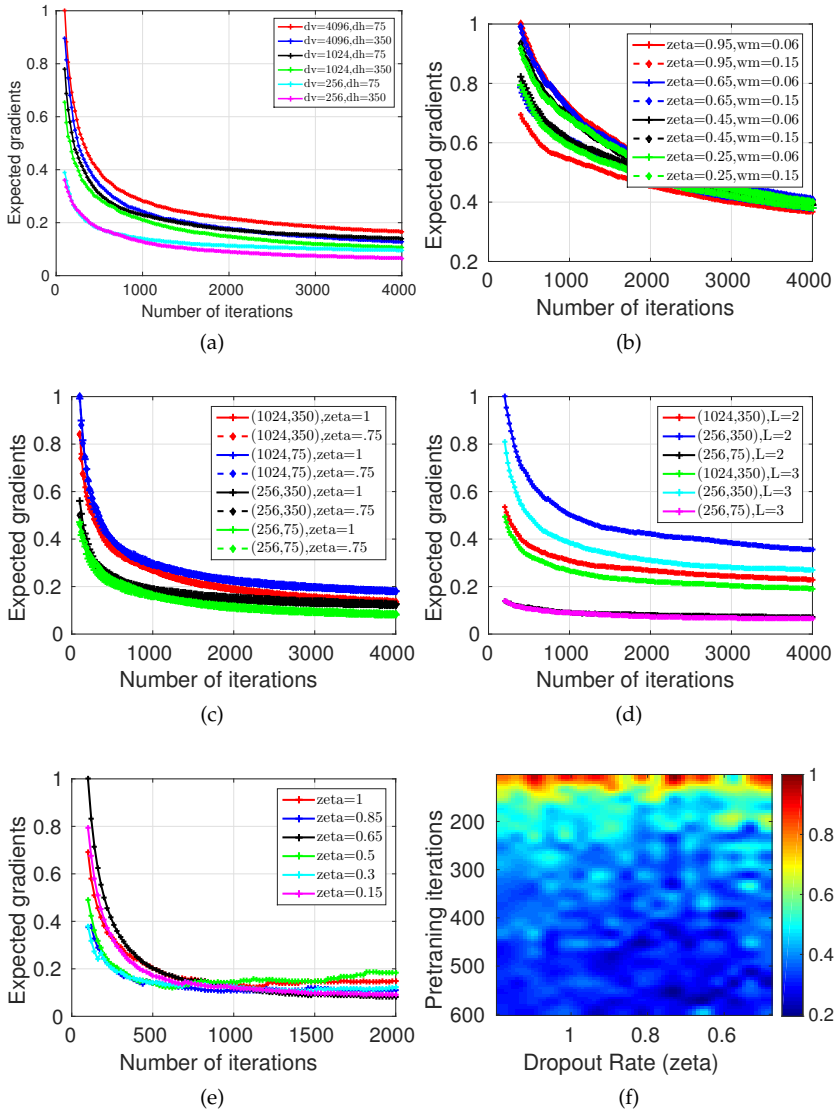


Figure 5.1: Decay of expected gradients (*MNIST*) versus network depth, hidden layer sizes and the influence of pretraining

for small w_{m}s. It is interesting that across all cases, the choice of ζs has almost a negligible influence. Figure 5.1(c), show the interaction of network sizes vs ζ, and as observed in Figure 5.1(a), the networks lengths dominate the convergence with the visible layer length being the most influential factor. The data moments of cifar are larger than mnist and neuro, and as suggested by (5.17)

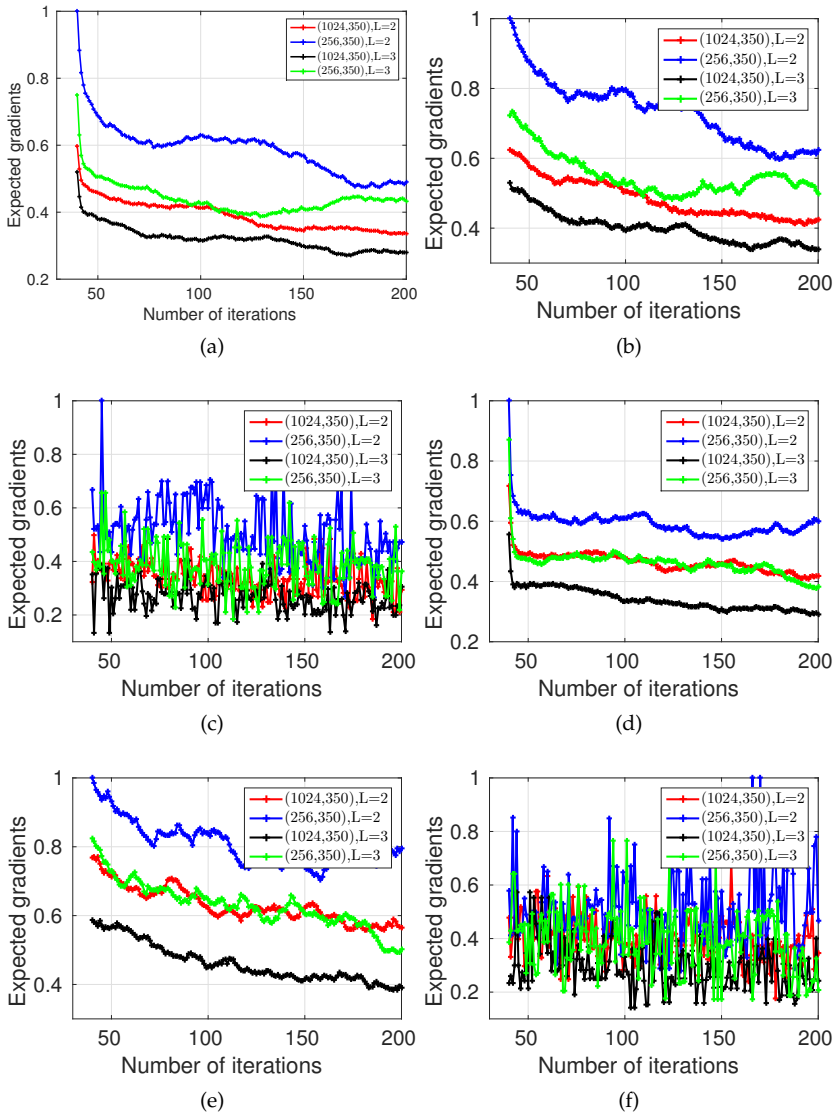


Figure 5.2: Choice of stopping distribution, and the influence of network depth and hidden layer sizes

and (5.18), this results in a stronger influence of ζ, w_{m} and the network sizes on the decay of gradients – which can be seen from Figures 5.1(b,c). Figure 5.1(d) shows the influence of the network depth with $w_{\mathrm{m}} = 1/\sqrt{\mathrm{d_v}\,\mathrm{d_h}}$ and $\zeta = 0.5$. Clearly, the expected gradients are influenced more by the layer lengths than the

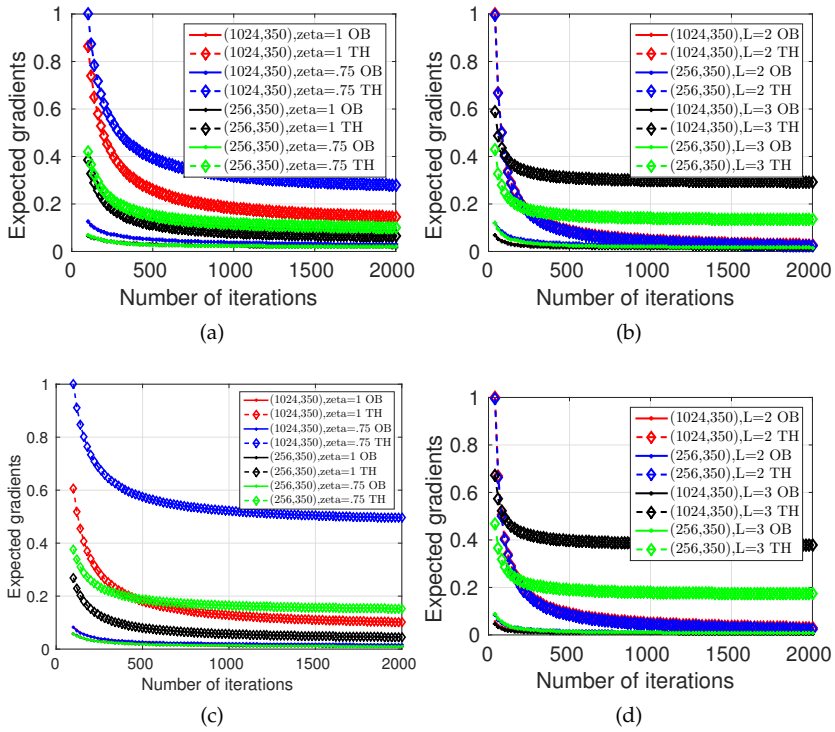


Figure 5.3: Predicted vs. observed trends of expected gradients

number of layers itself, as predicted by (5.22). Overall, convergence is faster for smaller and thinner networks with changes in visible layer length dominating the corresponding changes in the hidden layer lengths.

Does dropout compensate pretraining? Figure 5.1(e) shows the effect of changing the dropout rate in a 3-layered network. The influence of μ_x, τ_x and w_m were much stronger for caltech and cifar than the rest. Although the overall effect is small, the convergence is slower for very small (red curve) and very large (cyan, magenta curves) ζs (see (5.31)), and the trends are noisy for the neuro dataset. This is not surprising because, unlike the other computer vision datasets, neuro data includes co-registered features with very small data moments and small changes in feature values correspond to class difference. Figure 5.1(f) evaluates Theorem 5.16, by comparing pretraining (rows) to dropout (columns). Across all datasets, expected gradients clearly decrease as pretraining iterations increase, while for small and large dropouts,

the convergence is slower. The best regimes, as predicted by (5.31), are large pretraining with small dropout (bottom-right region on Figure 5.1(f) and $\zeta \approx 0.5$ with less pretraining (center of the image). Strong pretraining and dropout rate around 0.5 give fastest empirical gradients (dark blue region in Figures 5.1(f)) In summary, dropout rate has less influence and pretraining compensates dropout.

Influence of $\mathbb{P}_R(\cdot)$: The different choices of $\mathbb{P}_R(\cdot)$ used in Section 5.3 for Theorem 5.1 and Corollaries 5.3 and 5.4 are evaluated in Figure 5.2. The rows correspond to the four datasets while the columns correspond to using different $\mathbb{P}_R(\cdot)$. For the networks in the first column, R is uniformly distributed over the last 15 iterations. The second column used a Gamma distribution peaking at the last iteration and decreasing monotonically from the last to first iteration (an instance of the $\mathbb{P}_R(\cdot)$ used in (5.12)). The last column calculated expected gradients over a randomly chosen iteration from the last 15 iterations. Beyond the fact that the decay in the last column is noisier than the others, these types of plots are useful in choosing the best $\mathbb{P}_R(\cdot)$ from a dictionary of distributions. Overall, the plots show that the whenever $\mathbb{P}_R$ is chosen reasonably (first and second columns; third column is delta stopping distribution), the decay trends are consistent, as predicted by the analysis, ant the theoretical and empirical expected gradients decay trends are similar across several choices of hyperparameters and network architecture. Such plots can be used as surrogates for choosing when to stop the gradient updates – an alternative for measuring performance on some validation set.

Predicted versus observed trends: The discrepancy between the trends predicted by our results relative to the empirical observations are shown in Figure 5.3. Each row is a dataset. The first column shows pretraining and the second one corresponds to pretraining + supervised tuning. Clearly, the predicted trends (dotted lines) follow the observed ones (thick lines). Overall, across all the datasets and architectures, the predicted bounds from the results in the paper are larger than the observed ones, but the trends nevertheless are consistent. Also, there is a bias in the predicted bounds, coming from the terms (that do not contain N) involving the network degrees of freedom (i.e., network lengths). The estimate of their contribution is larger, suggesting that there is some redundancy.

Gradient norms and Local optima: Recall that it is NP Hard to check local optimality even for simple nonconvex problems (Murty and Kabadi, 1987; Pardalos and Schnitger, 1988). As this may be important for a practitioner, we address it empirically. Specifically, after performing N iterations (suggested by our theoretical results), we randomly sampled points around our final estimate $\mathbf{W}^{\mathrm{R}}$ (i.e., perturbed estimate). We observed that the fraction of such points that were better (by 0.02% of the objective value) than our estimate $\mathbf{W}^{\mathrm{R}}$ rapidly decreases as N increases. These fractions represent the probability that $\mathbf{W}^{\mathrm{R}}$ empirically satisfies first order local optimality conditions. Overall, these evaluations, when considered together with observations from (Bengio, 2009a; Erhan et al., 2010b; Baldi and Sadowski, 2014) provide empirical evidence that the results in Section 5.3–5.5 are sensible.

Visualizing trends

The interplay of all the structural and learning parameters shown by the decay bounds in (5.9), (5.18) and (5.22) are complicated to interpret towards designing new networks. Although the experiments presented above provide empirical support for these results, the decay trends were measured mainly as a function of N. To better understand the bounds, Figures 5.4–5.6 visualize the trends predicted. Note that here we simulate the bounds derived earlier in the technical results from Sections 5.3–5.6, unlike the results presented above which were based on the networks learned via Algorithm 2–4. Except Figures 5.4(b,d) and 5.6(b) which show the gradient iterations and optimal dropout rates respectively, all the other images are scaled to the corresponding maximum values since they are essentially computing the decay bound. The hyper-parameters N, B, γ and w are fixed for constructing Figures 5.4–5.6, and so, increasing N may resolve some of the outcomes/issues that will be pointed out.

1. Figure 5.4(a) shows the trivial trend where the decay bound increases as the input and output layer lengths increase. Small denoising rate i.e., large $\tilde{\zeta}$s in (5.18) lead to larger bounds, which is shown in Figure 5.4(c). Observe that for a reasonably small d_{x}, the influence of the denoising rate is almost negligible (top half of Figure 5.4(c)). The iteration estimates from Figures 5.4(b,d) are interesting. Recall the parameters ϵ and δ from Definition 5.5. With repeated runs of the random stopping stochastic gradients, $1-\delta$ denotes the probability that the maximum allowable norm of the gradients at the stopping iteration is ϵ. Figures 5.4(b,d) show

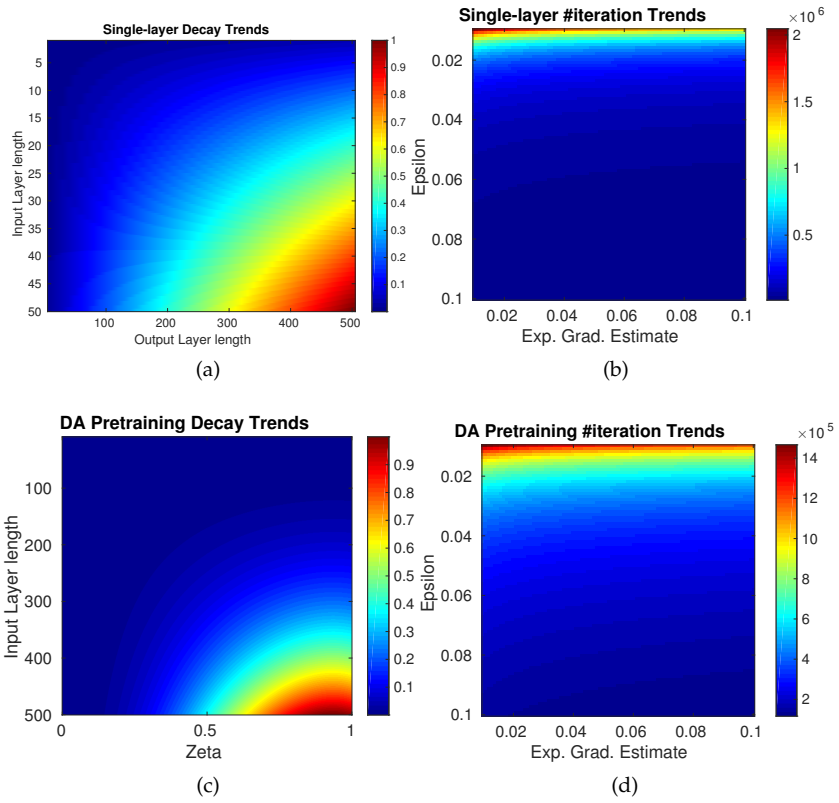


Figure 5.4: Decay trends of single-layer networks versus input and output layer lengths

that ϵ plays a far more influential role than δ. Specifically, this implies that even small changes in ϵ require adjusting N appropriately to maintain the same level of convergence.

2. The top and bottom rows in Figure 5.5 show the trends predicted by the bounds for non-dropout and dropout networks respectively. Figure 5.5(a) shows the strong influence of the network depth and the hidden layer lengths. This is not surprising from the structure of the bound in (5.22). Figure 5.5(b) clearly shows the necessity of pretraining, especially in networks with large layer lengths. It is interesting to see that large hidden layer lengths completely dominate any changes in the dropout rate (Figure 5.5(d)). Figure 5.5(e) shows that the dropout rate has a much smaller influence than the pretraining goodness (see discussion below

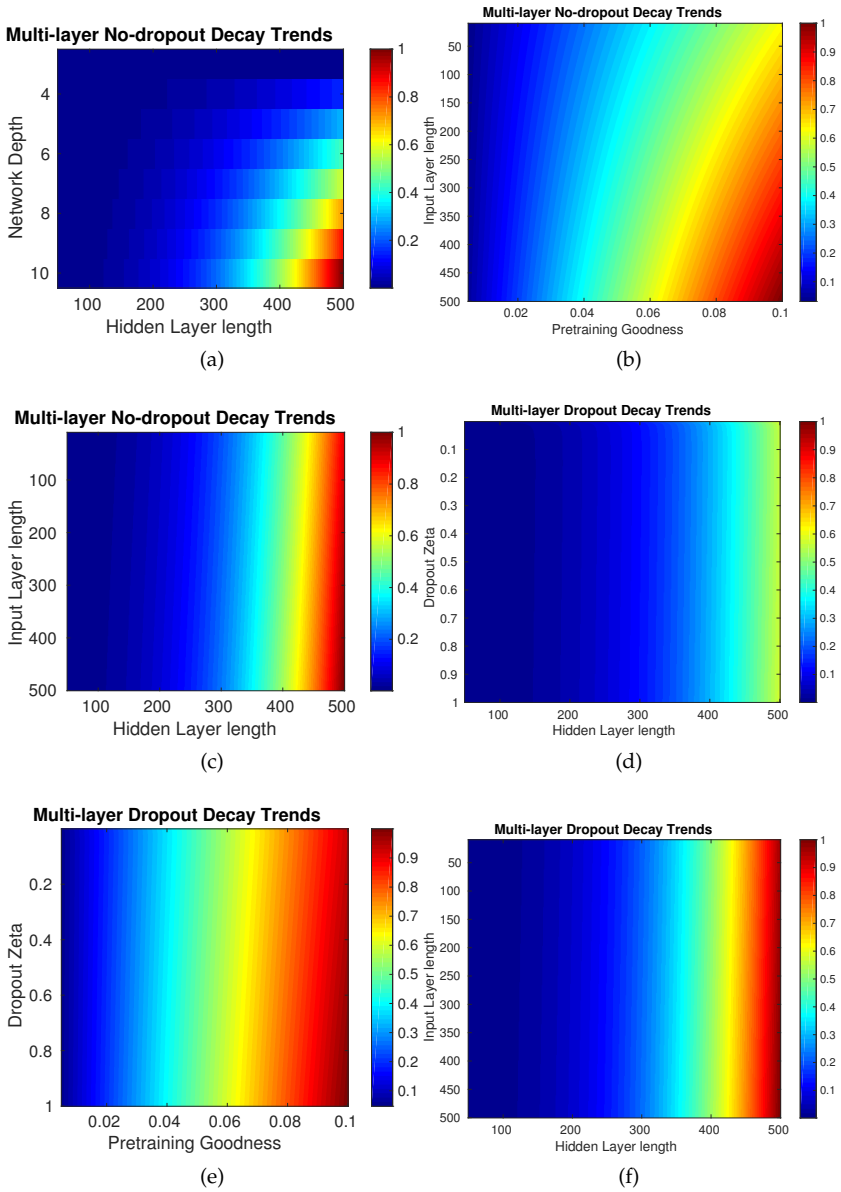
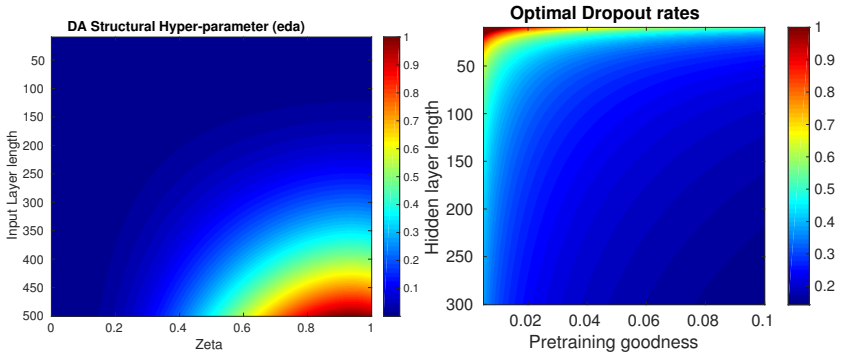


Figure 5.5: Decay trends of multi-layer networks versus input and output layer lengths

about Figure 5.6(b)). Figures 5.5(c,f) show that hidden layer lengths are more influential than d_0. This may be seen from (5.22) and (5.31) because



(a) (b)

Figure 5.6: Denoising and dropout Trends versus expected gradients

the depth parameter L inherently puts more weight on the hidden layer constants e_l^m.

3. The optimal denoising rate will be given by the solutions of (5.17). Figure 5.6(a) plots e^{da} as the denoising rate (x-axis) and d_x (y-axis) are changed. $\tilde{\zeta}$ smaller than 0.5 seems to be the 'good' regime, however, d_x has a much stronger influence. For very large d_x, the ranges of e^{da} do not seem to be influenced by ζ (see the top half vs. bottom half of Figure 5.6(a)). This is in agreement with the interpretations made earlier in Section 5.4.
4. Figure 5.6(b) shows the optimal dropout rate ζ (from (5.33)) as α_ls and d_ls are changed appropriately. The interpretations made earlier in the remarks of Corollary 5.18 can be seen here, e.g., ζ is small whenever α_ls is large. Further, (5.33) seems to indicate that very large hidden layer lengths need large dropout rates independent of how good the pretraining is. If one marginalizes the influence of pretraining (i.e., summing up columns of Figure 5.6(b)), with no extra information of the network depth and also, the layer lengths are reasonable (or large), the *go to* choice of dropout is simply 0.5. The authors in (Baldi and Sadowski, 2014) proved that 0.5 dropout rate is optimal for dropout dynamics, and Figure 5.6(b) shows that such a rate is really the default choice even from the perspective of convergence. While presenting design strategies in Section 5.8, we show that other choices are optimal when extra information about data and network structure are allowed to be used.

5.8 Discussion

Building upon the evaluations in Section 5.7 where the technical results from Sections 5.3–5.5 are assessed empirically, we now turn to the design choice problem presented in Section 5.1 for constructing design strategies or *ground rules* to drive the construction and learning of deep networks. We approach this problem in a few different ways – motivated mostly by the arguments in Section 5.1. Once the network structure is set up, we then compute the minimum required number of training instances and the batchsize using other learning hyper-parameters and input data statistics – all of which come from Theorems 5.8, 5.11, 5.15, Corollaries 5.13, 5.18, 5.19, and the corresponding discussion in Sections 5.4, 5.5, 5.5, 5.6 and 5.6. Corollary 5.19 guides the overall design procedure.

Revisiting the design choice problem

By design of deep network, we mean the task of choosing all the network hyper-parameters (see Section 5.1 for a discussion). For convenience, we first list the key hyper-parameters involved in different stages of the network design in Table 5.1. The pretraining hyper-parameters (the denoising rates for the $L-1$ DAs, the pretraining batchsize and iterations) are not included in this list because they are inherently controlled by α_ls. Once α_ls are chosen, these can be picked appropriately based on the discussion in Section 5.4. Depending on the task at hand, we are at liberty to change the output layer length d_L, if needed. Specifically, for classification or regression on a *fixed* set of outcomes $\mathbf{y}$, it is reasonable to fix d_L together with d_0. On the other hand, when the networks are used for learning representations, allowing d_L to be chosen by the design procedure is more sensible. So, we include d_L within the design choice as well.

The motivation in Section 5.1, especially the multiple sites setup in Section 5.1, deals with the comparability of different network designs. We make a few remarks about the criteria and the availability (or the lack thereof) of unsupervised data before describing the procedure. We see from empirical observations that $\mathbf{W}$s can be interpreted as "filters" transforming the given signals. For instance, the transformation maps in deep Boltzmann machines can be interpreted as representation 'stubs' which the network tries to identify in the inputs, e.g., the binary $\mathbf{W}$s learned with MNIST data (Nair and Hinton, 2010). In convolutional networks, $\mathbf{W}$s directly correspond to the filter maps that

Table 5.1: The hyper-parameters involved in designing deep network

Structural Hyper-parameters	
L	Network depth
$d_0, \ldots, d_L$	Input, hidden and output layers length

Learning Hyper-parameters	
$\gamma^1, \ldots, \gamma^L$	Stepsize choices
$w_m^1, \ldots, w_m^L$	Box-constraints for each DA
$\zeta_1, \ldots, \zeta_{L-1}$	Dropout rates for $L-1$ hidden layers
$\lceil \frac{1}{\kappa} \rceil$	Number of epochs
$N_u^1, \ldots, N_u^{L-1}$	$L-1$ DA iterations
N_s	Backpropagation iterations
S_u and S_s	Unlabeled and Labeled instances

Goodness Criteria	
$(\alpha_1, \delta_{\alpha_1}), \ldots (\alpha_{L-1}, \delta_{\alpha_{L-1}})$	Pretraining goodness for $L-1$ hidden layers
$\Psi_1, \ldots, \Psi_{L-1}$	Convergence goodness (Non-dropout network)
$\Phi_1, \ldots, \Phi_L$ and Φ_f	Convergence goodness (Dropout network)
(ϵ, δ)	Goodness of solution/estimate

are sequentially applied on the transformed inputs (Kavukcuoglu et al., 2010; Lee et al., 2009). Therefore, it is reasonable that we ask that the dimensions of these filters $\mathbf{W}$s or hidden layer lengths d_ls be the same to ensure that networks are comparable via the transformation maps. We refer to this setting by saying that the network design needs to be "transferable". Alternatively, if one is more concerned with generalization, then such a restrictive control on d_ls is not necessary, and in certain cases may also lead to sub-optimal networks. Further, for each of these settings, depending on the availability of unlabeled data, the multi-layer RSG in Algorithm 4 may either include a layer-pretraining procedure or may not. So, we have a total of four different design procedures.

At a high level, the convergence and pretraining goodness criteria are used to bound the hidden layer lengths (using input and output layers), which are then used to decide the learning hyper-parameters. For convenience, the constraints from (5.24) and (5.35) are summarized in Table 5.2 (see the remarks for Corollaries 5.13 and 5.19). Based on Table 5.2, Tables 5.3 presents the design procedure for datasets with unlabeled data whereas Table 5.4 presents the fully-supervised design. The inputs to Tables 5.3 and 5.4 are the pretraining and convergence goodness criteria (see Table 5.1), the desired depth of the network, number of epochs allowed and the input data. The specific steps within these

procedures are self-explanatory – using these inputs, the procedures first decide the network structure followed by the appropriate learning hyper-parameters. Wherever appropriate, some free hyper-parameters like γ and w_l^ms are listed in Tables 5.3 and 5.4 which can be set to reasonable values. Note that we do not claim that these strategies are ideal – instead, they are the *simplest* ones directly implied by our results with no additional assumptions.

Table 5.2: Design constraints from (5.24) and (5.35) in Corollaries 5.13 and 5.19

U.1	$\log(\zeta) + \log(d_0) + \log(d_1) + \log(d_2)$	$= \log(\frac{1}{\Psi_1 \alpha_1})$
U.2	$\log(\zeta) + \log(d_1) + \log(d_2) + \log(d_3)$	$= \log(\frac{1}{\Psi_2 \alpha_2})$
$\vdots$	$\vdots$	$\vdots$
U.L-2	$\log(\zeta) + \log(d_{L-3}) + \log(d_{L-2}) + \log(d_{L-1})$	$= \log(\frac{1}{\Psi_{L-2} \alpha_{L-2}})$
U.L-1	$-\log(\zeta) + \log(d_{L-1}) + \log(d_L)$	$= \log(\frac{1}{\Phi_L})$
U.L	$-2\log(\zeta) + \log(d_L)$	$= \log(\frac{1}{\Phi_f})$
S.1	$\log(\zeta) + \log(d_0) + \log(d_1) + \log(d_2)$	$= \log(\frac{1}{\Phi_1})$
S.2	$\log(\zeta) + \log(d_1) + \log(d_2) + \log(d_3)$	$= \log(\frac{1}{\Phi_2})$
$\vdots$	$\vdots$	$\vdots$
S.L-2	$\log(\zeta) + \log(d_{L-3}) + \log(d_{L-2}) + \log(d_{L-1})$	$= \log(\frac{1}{\Phi_{L-2}})$
S.L-1	$-\log(\zeta) + \log(d_{L-1}) + \log(d_L)$	$= \log(\frac{1}{\Phi_L})$
S.L	$-2\log(\zeta) + \log(d_L)$	$= \log(\frac{1}{\Phi_f})$

Interpreting and Choosing α_ls, δ_{α_l}s, ϵ and δ

These goodness criteria influence the large deviation estimates of the pretraining and dropout learning and their choice is vital towards the structure of the network. Recalling the definition of (ϵ, δ)-solution from Definition 5.5, and further using Theorem 5.11, we have the following constraints on these criteria.

$$\begin{aligned} &0 < \alpha_l < 1, \quad 0 < \delta_{\alpha_l} < 1 \quad \text{for} \quad l = 1, \ldots, L-1 \\ &0 < \epsilon < 1, \quad 0 < \delta < 1 \end{aligned} \tag{5.38}$$

$(\alpha_1, \delta_{\alpha_1}), \ldots, (\alpha_{L-1}, \delta_{\alpha_{L-1}})$ determine the large deviation pretraining estimates of the $L-1$ hidden layers – see (5.23), Theorem 5.11, Theorem 5.16 and the corresponding remarks. (ϵ, δ) governs the large deviation estimate of the

Table 5.3: Deep network design – S_u and S_s available

Given the goodness criteria α_ls, δ_{α_l}s, Ψ_ls, Φ_L, Φ_f, ϵ and δ from Table 5.1
depth L, input length d_0, number of epochs $\lceil \frac{1}{\kappa} \rceil$

If d_L and ζ are not fixed
If d_ls need to be transferable, use $d_L = \lceil \frac{65}{4\alpha_{L-1}} \rceil$;
else pick any $d_L < \lceil \frac{65}{4\alpha_{L-1}} \rceil$;
solve the linear system **U.1, …, U.L** from Table 5.2
If either d_L or ζ is given/fixed
solve the linear system **U.1, …, U.L** from Table 5.2
If d_L and ζ are fixed
solve the linear system **U.1, …, U.L-1** from Table 5.2

Pretraining: For each of the $l = 1, \ldots, L-1$ DAs
Select denoising rates, w_m and stepsizes appropriately (No restriction)
Compute N_u^l from (5.21) for $(\alpha_l, \delta_{\alpha_l})$ solution

Backprop using Algorithm 4:
Compute $e_1^m, \ldots, e_L^m$
Choose γ and $w_1^m, \ldots, w_L^m$ appropriately (No restriction)
If $\zeta \approx 0$ or ≈ 1
use (5.33) with $d = \frac{1}{L-1} \sum_{l=1}^{L-1} d_l$, or simply set $\zeta = 0.5$
Estimate required N_s using (5.32) for (ϵ, δ) solution

supervised backpropagation. Since these four sets of criteria come from large deviation estimates (see Definition 5.5), these are straightforward to setup:

- Choose α_ls and ϵ such that $0 < \alpha_l, \epsilon \ll 1$
- Choose δ_{α_l}s and δ such that $0 \ll \delta_{\alpha_l}, \delta < 1$

Interpreting and Choosing Ψ_ls, Φ_ls and Φ_f

We find that Ψ_ls, Φ_ls and Φ_f govern the contribution of each of the 1-NNs that compose the L-NN to the decay of projected gradients – see Corollaries 5.13, 5.19 and the corresponding remarks. Unlike the goodness criteria from Section 5.8 which are directly related to the overall large deviation gradient norm of the network (and so easier to interpret and choose), the meta-parameters Ψ_ls, Φ_ls and Φ_f are not straight forward to set up. They come out of the terms in the decay bounds – see (5.24) and (5.35) – and one will want them to be as small as possible which in turn leads to small expected gradients. Using (5.24) and

Table 5.4: Deep network design – S_u not available

Given the goodness criteria Φ_ls, Φ_f, ϵ and δ from Table 5.1
depth L, input length d_0, number of epochs $\lceil \frac{1}{\kappa} \rceil$

If d_L and ζ are not fixed
If d_ls need to be transferable; pick $\zeta = 0.5$
else choose any $0 < \zeta < 1$
solve the linear system **S.1, ..., S.L** from Table 5.2
If either d_L or ζ is given/fixed
solve the linear system **S.1, ..., S.L** from Table 5.2
If d_L and ζ are fixed
solve the linear system **S.1, ..., S.L-1** from Table 5.2

Backprop using Algorithm 4:
Compute $e_1^m, \ldots, e_L^m$. Estimate α_ls with random network initialization
Choose γ and $w_1^m, \ldots, w_L^m$ appropriately (No restriction)
If $\zeta \approx 0$ or ≈ 1
use (5.33) with $d = \frac{1}{L-1} \sum_{l=1}^{L} d_l$ and $\alpha = \frac{1}{L-1} \sum_{l=1}^{L-1} \alpha_l$
or simply set $\zeta = 0.5$
Estimate required S_s using (5.32) for (ϵ, δ) solution

(5.35) have,

$$\begin{aligned} \Psi_l > 0 \quad &\text{for} \quad l = 1, \ldots, L-1 \\ \Psi_l > 0 \quad &\text{for} \quad l = 1, \ldots, L \quad \text{and} \quad \Psi_f > 0 \end{aligned} \tag{5.39}$$

Recall that given Ψ_ls, Φ_ls and Φ_f, the network is guaranteed to achieve the convergence level from (5.25) and (5.36). A simple way to interpret them is by thinking of them, loosely, as *significance levels*. If the pre-specified threshold is large, the test is liberal and on the other hand, a smaller threshold makes the test very conservative. Too small values for Ψ_ls, Φ_ls and Φ_f lead to large networks, which requires large N, batchsize and sample sizes (too liberal). On the other hand, we also want to avoid prohibitively smaller networks that may not generalize (too conservative). Since Ψ_ls, Φ_ls and Φ_f are proportional to $\frac{1}{d^{-3}}$ where d denotes the average hidden layer length (see Table 5.1), their *typical* ranges seems to be closer to 0, and therefore, without loss of generality, we restrict ourselves to $[0, 0.01]$.

- $\mathbf{0 < \Psi_l, \Phi_l, \Phi_f \ll 1}$**:** The smaller these criteria are, the larger the hidden and output layer lengths will be, which directly follows the equalities

U.l and **S.l** (or (5.24) and (5.35)). Following the overwhelming evidence for large hidden layers (Bengio, 2009a; Hinton, 2010; Bengio, 2012), one may clearly choose Ψ_l, Φ_ls and Φ_f to be small ($\to 0$). Such small values will make the network *liberal* i.e., given sufficient training data and N, the network may model any complex concept but will be computationally very expensive.

- $\mathbf{0 \ll \Psi_l, \Phi_l, \Phi_f < 1}$**:** However, Corollaries 5.14 and 5.20 showed that the taller networks may not always generalize better than shorter ones. In this alternate regime where the Ψ_l, Φ_ls and Φ_f are reasonably larger than 0, the hidden layer lengths may be small i.e., the network is *conservative* and can achieve faster convergence with small training size and N but may not be able to learn complex concepts.
- $\mathbf{\Psi_l \alpha_l \approx \Phi_l}$ As described earlier, α_ls are easier to choose compared to Ψ_ls, Φ_ls and Φ_f. Further, the linear systems, **U.l**s and **S.l**s from Table 5.1 suggest an interesting relation between these criteria
$$\alpha_l \Psi_l \approx \Phi_l \quad l = 1, \dots, L-2 \tag{5.40}$$
Since $\alpha_l \ll 1$, the good regimes of Φ_ls are much smaller than Ψ_ls.

This list of regimes gives a clear strategy to choose the appropriate Ψ_ls and Φ_ls. Using these prototypical ranges for α_ls, Ψ_ls, Φ_ls and Φ_f, we present a few example designs in the next section to demonstrate the practicality of our design procedures from Tables 5.3 and 5.4,

Example designs:

1. **Example 1.** $d_0 = 100$, 10 *epochs*, $L = 5$, *No* S_u*:* Designs from Table 5.4 are applicable here.
   - $d_L = 5$, $\zeta = 0.5$*:*
   $$\begin{aligned} &\Phi_1 = \Phi_2 = \Phi_3 = 10^{-5} \quad \Phi_5 = 10^{-3} \\ &\implies d_1 = 100, d_2 = 20, d_3 = 100, d_4 = 100 \end{aligned} \tag{5.41}$$
   - $d_L = 5$*:*
   $$\begin{aligned} &\Phi_1 = \Phi_2 = \Phi_3 = 10^{-5} \quad \Phi_5 = 10^{-3} \quad \Phi_f = 0.01 \\ &\implies \zeta = 0.25, d_1 = 50, d_2 = 80, d_3 = 100, d_4 = 50 \end{aligned} \tag{5.42}$$
   - *No* d_L*, No* ζ*:*

$$\begin{aligned}&\Phi_1 = \Phi_2 = \Phi_3 = 10^{-5} \quad \Phi_5 = 10^{-3} \quad \Phi_f = 0.01\\ &\implies \text{with } \zeta = 0.5, \quad d_1 = 50, d_2 = 80, d_3 = 100, d_4 = 50, d_5 = 25\end{aligned} \tag{5.43}$$

2. **Example 2.** $d_0 = 100$, 10 *epochs*, $L = 5$*:* The designs from Table 5.3 are applicable here.
   - $d_L = 5$, $\zeta = 0.5$*:*

$$\begin{aligned}&\Psi_1 = \Psi_2 = \Psi_3 = 10^{-3} \quad \alpha_1 = \alpha_2 = \alpha_3 = 10^{-2} \quad \Phi_5 = 10^{-3}\\ &\implies d_1 = 100, d_2 = 20, d_3 = 100, d_4 = 100\end{aligned} \tag{5.44}$$

   - $d_L = 5$*:*

$$\begin{aligned}&\Psi_1 = \Psi_2 = \Psi_3 = 10^{-3} \quad \alpha_1 = \alpha_2 = \alpha_3 = 10^{-2}\\ &\Phi_5 = 10^{-3} \quad \Phi_f = 0.01\\ &\implies \zeta = 0.25, d_1 = 50, d_2 = 80, d_3 = 100, d_4 = 50\end{aligned} \tag{5.45}$$

   - *No* d_L*, No* ζ*:*

$$\begin{aligned}&\Psi_1 = \Psi_2 = \Psi_3 = 10^{-3} \quad \alpha_1 = \alpha_2 = \alpha_3 = 10^{-2}\\ &\Phi_5 = 10^{-3} \quad \Phi_f = 0.01\\ &\implies \text{with } \zeta = 0.5, \quad d_1 = 50, d_2 = 80, d_3 = 100, d_4 = 50, d_5 = 25\end{aligned} \tag{5.46}$$

3. **Example 3.** $d_0 = 100$, 10 *epochs*, $L = 5$, *No dropout:* Designs from Table 5.3 are applicable here with $\zeta = 1$.
   - $d_L = 5$*:*

$$\begin{aligned}&\Psi_1 = \Psi_2 = \Psi_3 = 10^{-4} \quad \alpha_1 = \alpha_2 = \alpha_3 = 10^{-2} \quad \Phi_5 = 10^{-3}\\ &\implies d_1 = 200, d_2 = 50, d_3 = 100, d_4 = 200\end{aligned} \tag{5.47}$$

   - *No* d_L*:*

$$\begin{aligned}&\Psi_1 = \Psi_2 = \Psi_3 = 10^{-4} \quad \alpha_1 = \alpha_2 = \alpha_3 = 10^{-2}\\ &\Phi_5 = 10^{-3} \quad \Phi_f = 0.1\\ &\implies d_1 = 100, d_2 = 100, d_3 = 100, d_4 = 100, d_5 = 10\end{aligned} \tag{5.48}$$

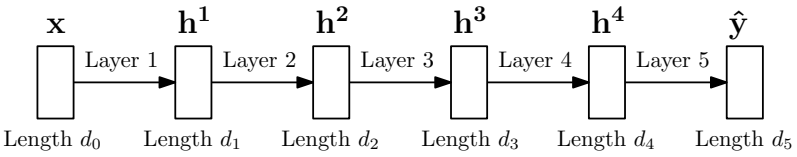
Note that the latter half of the design procedure (from Tables 5.3 and 5.4) is not shown in these examples. The examples here assume that some of the Φ_ls and Ψ_ls are the same, although there is no such restriction. Specifically,

the ranges of Φ_L and Φ_f play a role in deciding the ranges of higher layer lengths. Figure 5.7 shows more evidence of this behavior including some good strategies to choose Φ_L and Φ_f based on (the rest of) the criteria. The designs in Figure 5.7 construct a network of depth $L = 5$ (shown in the top row), with $\Phi_5 = \sqrt{\Phi_1}$ (Figures 5.7(b-d)) and $\Phi_5 = \sqrt[3]{\Phi_1}$ (Figures 5.7(e-g)). When $\Phi_5 = \sqrt[3]{\Phi_1}$ the hidden layer lengths have high variance compared to $\Phi_5 = \sqrt{\Phi_1}$. This variance is much higher when d_5 (the output layer length) is fixed while allowing ζ to change (see Figures 5.7(c,f)). Note that the x-axis in the plots shows $\Phi_1 = \Phi_2 = \Phi_3$, and y-axis gives layer lengths in $\log_{10}$ scale.

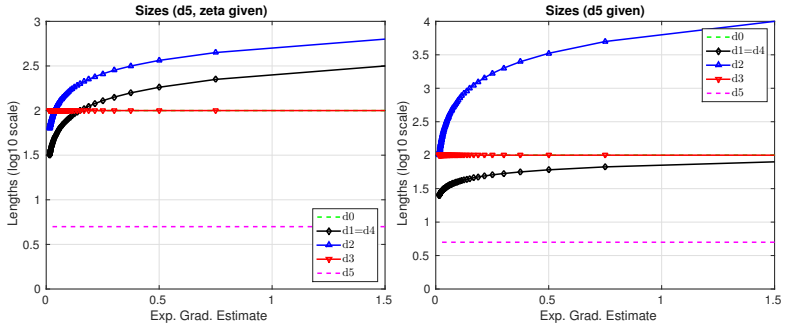
Multi-center Studies: Network design, Resource Allocation and Savings

The plots in Figure 5.7 and the network lengths in Section 5.8 show some evidence of how the designs in Table 5.3 and 5.4 may be deployed in practice. Building upon these results, we now tackle the design choice problem for ADNI data considered earlier in Section 5.7. We first construct several sets of designs for this dataset – both transferable and not (see Section 5.8), perform learning and then compare their expected gradients trends to assess the usefulness of the proposed design procedures. Once this is done, we then evaluate the multi-center setting described in Section 5.1 where multiple sets of learning models that are comparable – are constructed on different (but related) datasets across multiple sites. Specifically, for a two-center setting we present resource savings using the proposed design procedures.

Setup: For a given set of goodness criteria (similar to those used in Example 2, Section 5.8), three different deep networks are designed using Tables 5.3 and 5.4 respectively. These three designs correspond to a transferable design (i.e., hidden layer lengths are same). Once the designs are computed, each is then trained using 10 different stopping iterations (each sampled uniformly from the last 50 iterations). This leads to 20 different networks for one set of goodness criteria – a total of 5 different sets of goodness criteria are used. Note that all these 100 learned networks are trained on the *same* ADNI dataset. The goal then is to quantify the discrepancy within and across these 5 sets of deep networks. Clearly, for a fixed goodness criteria, we expect the designs to achieve the same level of empirical expected gradients and similar generalization. On

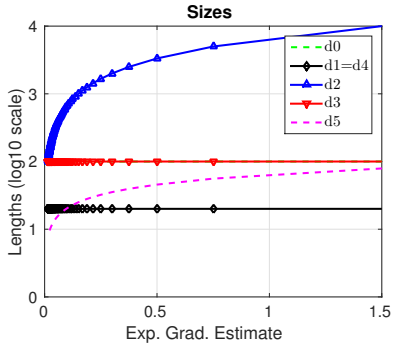


(a)

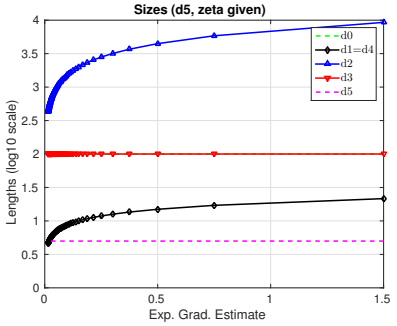


(b)

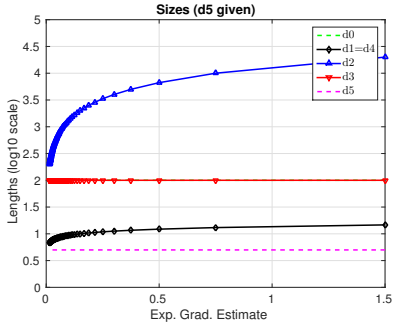
(c)



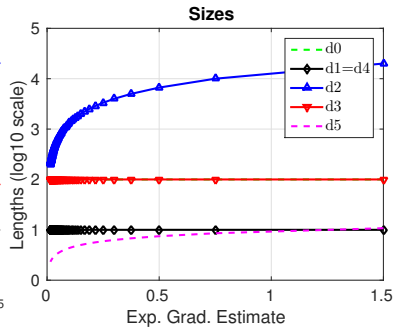
(d)



(e)



(f)



(g)

Figure 5.7: Example Designs (Network Lengths)

the other hand, we expect the empirical parameter convergence to show some interesting trends as the goodness criteria vary.

Comparing designs with a *given* goodness criteria

The simplest choice for a direct discrepancy measure comparing two models is to compute the test set error (generalization), or some form of a stability measure as reported in (Hardt et al., 2015). The design procedures in Tables 5.3 and 5.4 are entirely motivated by parameter convergence, which is neither quantified by the test set error, nor the empirical stability. Also the empirical stability computes the difference in the risk computed on two datasets that differ at most in one instance, whereas we are interested in comparing networks which learn data generated from the same underlying distribution. Hence we use the following measures.

(D1) the difference in the learning objective (or loss)
Since the goal is generalization in the context of parameter convergence, quantifying the absolute difference of the learning objective is a reasonable measure to evaluate

(D2) the norm of the difference of gradients at the stopping iteration
Because of the use of stochastic gradients, it is sensible to quantify the discrepancy between networks by comparing the parameter neighborhood at the stopping iteration. If such neighborhoods are similar between the replicates, then the estimates converged to 'similar' regions of the parameter spaces. The norm of the difference in gradients evaluates this measure which is only applicable for the transferable design where the network structure (and the number of parameters) is the same across designs.

For certain networks, computing **(D2)** is cheaper than **(D1)**. As pointed out earlier in the results from Sections 5.3–5.6, any lower bound on N, will lead to a lower bound on the training time, which then leads to a sense of generalization achievable such that overfitting is avoided. This is supported by (Hardt et al., 2015), and other empirical studies that have exhaustively studied stochastic gradients in deep networks (Bottou, 2010; Dauphin et al., 2014). Hence, these discrepancy measures will be evaluated against the gradient iteration count N.

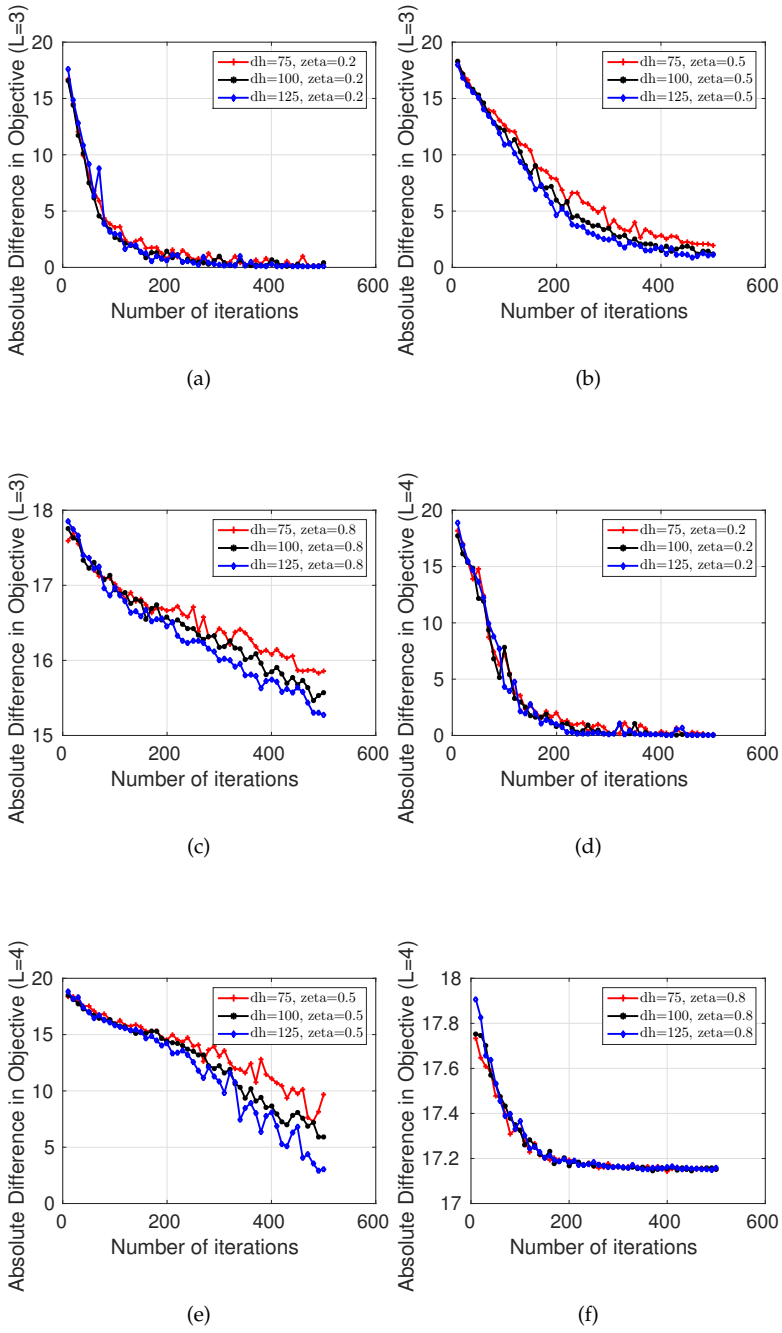


Figure 5.8: Difference in objective (Transferable design)

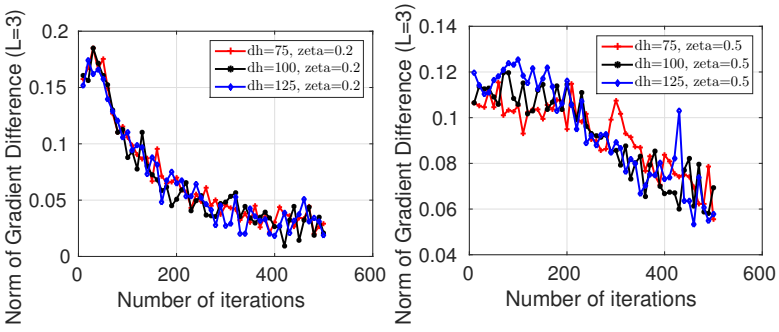


Figure 5.9: Difference in Gradients (Transferable design)

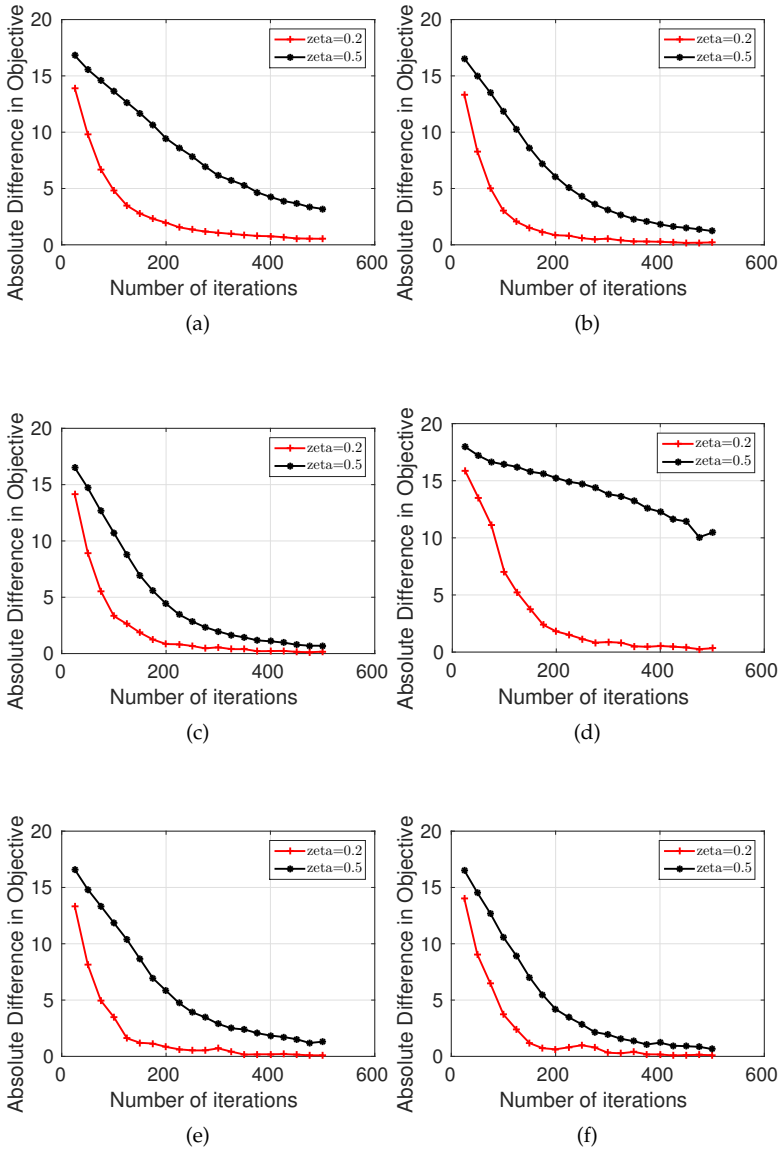


Figure 5.10: Difference in objective (Non-transferable design)

Results

Figures 5.8 and 5.9 show the results of transferable designs (same network structure) for depth 3 and 4, and Figure 5.10 shows the case where hidden layer lengths are not transferable allowing for fatter and thinner networks for the same depth. Unlike the plots in Section 5.7, the y-axes in these plots are not scaled and the absolute difference is presented.

- **Same hidden layer designs:** Figure 5.8 shows the average difference of the learning objective across the two sets of networks, while Figure 5.9 shows the norm of difference in gradients. Clearly, these discrepancy measures decrease as the iterations N increase. However, as shown by Figures 5.8(c,f) compared to Figures 5.8(a,b,d,e), the objective at stopping iterations are different whenever the dropout rate is small (i.e., large ζ) – an indicator of overfitting. Larger depth networks (bottom row versus the top row in Figure 5.8) seem to have a higher disagreement among the estimates learned, which is expected because the iterations needs to be increased as network depth increases. The norm of difference in gradients shown in Figure 5.9 seems to decrease as N increases, but saturates for certain designs (Figure 5.9(e)).
- **Different hidden layers designs:** A similar set of trends are observed for non-transferable designs from Figure 5.10 which permit differences in hidden layer lengths. Note that a total of 10 different sets of networks are designed for each set, each of which is learned with 10 stopping iterations i.e., the comparisons in Figure 5.10 are between two sets of 100 learned networks. The dropout rate still seems to play a key role, with differences between the two sets of networks increasing as the dropout rate decreases (red versus black curves). As observed earlier, the bottom row with ($\mathrm{L} = 4$) has larger discrepancy measures compared to the top row ($\mathrm{L} = 3$).

Although for the networks used in Figures 5.8–5.10 the dropout rate and the output length are fixed, a similar set of trends can be produced any other alternate designs from Tables 5.3 and 5.4.

Two-center resource savings

Building upon the validation experiments in Sections 5.8 and 5.8, we now look at the two center scenario where the goal is to design multiple *comparable* models for datasets coming in from two sites – see the discussion from Section 5.8. These two centers may correspond to two enrichment sites for a RCT –

and in each site we intend to design an optimal enrichment model using deep networks. Figure 5.11 shows the two center scenario. Whenever the two set of models $M_1^1, \ldots, M_G^1$ and $M_1^2, \ldots, M_H^2$ are designed using Table 5.3 and 5.4, the corresponding discussion and results from Section 5.8 ensure that they are comparable.

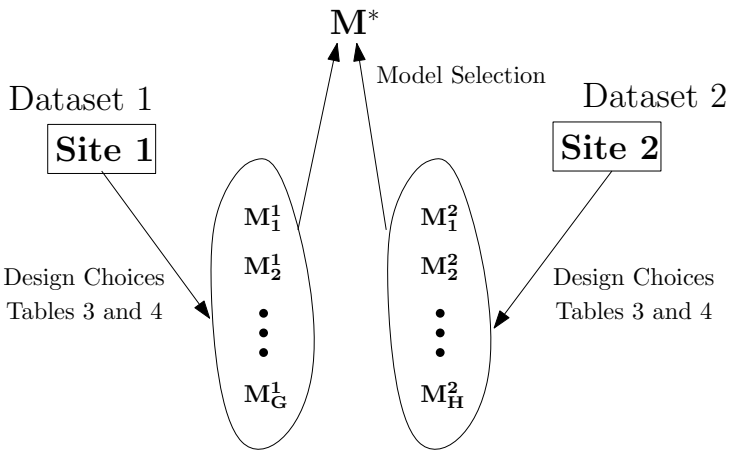


Figure 5.11: Two-center learning of deep networks

To see the resource benefits assume that $M_1^1, \ldots, M_G^1$ and $M_1^2, \ldots, M_H^2$ are rich with large hidden layer lengths and large depth. Tables 5.3 and 5.4 provide the smallest possible designs for the given goodness criteria. Specifically, this is either in terms of the size of the network (i.e., number of parameters to be learned) which will influence storage/memory requirements, or the minimum number of iterations required which summarizes the cost (and time) allocated to learn the models. To quantify this resource saving aspect of the design presented here, consider the setting where the models are learned on a workstation or a cloud platform. Every "compute hour" or calculation has a financial cost associated with it. Even in the modest setting where, for a given set of goodness criteria, the design choices reduce the number of computations for learning the models by 10%, and the dollar cost is say $\$1$ per hour, the computational savings can be substantial. For example, if $G = H = 50$, the financial resources need to be allocated to learn the models is reduced by at least a factor of 90. Although a more systematic empirical study is required to further explore

this aspect of savings in resource allocation, the above hypothetical example provides some justification for the applicability of Tables 5.3 and 5.4 from the monetary perspective.

5.9 Proofs

In this section, we will present the proofs for all the technical results discussed in the chapter.

Proof of Theorem 5.1

This proof for the 1-layer NN serves as a template for the results corresponding to DA and multi-layer NN. The proof relies on two inequalities concerning the behaviour of the noisy gradients. We first derive these inequalities.
Let $\delta = G(\eta; \mathbf{W}) - \nabla_{\mathbf{W}} f(\mathbf{W})$, where by definition

$$f(\mathbf{W}) = \mathbb{E}_\eta \mathcal{L}(\eta; \mathbf{W}) \qquad G(\eta; \mathbf{W}) = \frac{1}{B} \sum_{i=1}^{B} g(\eta; \mathbf{W}) \qquad g(\eta; \mathbf{W}) = \nabla_{\mathbf{W}} \mathcal{L}(\eta; \mathbf{W}) \tag{5.49}$$

Recall that in the single layer NN case $\eta = \{\mathbf{x}, \mathbf{y}\} \sim \mathcal{X}$. The Lipschitz constant of the activation function $\sigma(\cdot)$ is $\frac{1}{4}$ following the fact that $\{\sigma(\cdot)(1 - \sigma(\cdot))\} \leqslant \frac{1}{4}$. The first inequality that we derive computes an upper bound on the variance of noisy gradients $g(\eta; \mathbf{W})$ i.e., $\mathbb{E}_\eta \|g(\eta; \mathbf{W}) - \nabla_{\mathbf{W}} f(\mathbf{W})\|^2$. The second inequality quantifies the Lipschitz continuity of $\nabla_{\mathbf{W}} f(\mathbf{W})$ using continuity and boundedness properties of $\sigma(\cdot)$.
For the first inequality, denote δ to be the length $d_x d_y$ vector with $\delta_{ij} = g^{ij}(\eta; \mathbf{W}) - \nabla_{W_{ij}} f(\mathbf{W})$ where $i = 1, \ldots, d_y$ and $j = 1, \ldots, d_x$. Hence

$$\mathbb{E}_\eta \|\delta\|^2 = \mathbb{E}_\eta \sum_{ij} |\delta_{ij}|^2 = \sum_{ij} \mathbb{E}_\eta \left(g^{ij}(\eta; \mathbf{W}) - \nabla_{W_{ij}} f(\mathbf{W}) \right)^2 \tag{5.50}$$

where $g^{ij}(\eta; \mathbf{W})$ is noisy gradients with respect to W_{ij}. We upper bound this by computing the maximum variance of $g^{ij}(\eta; \mathbf{W})$s.
For the 1-layer NN, $\mathcal{L}(\eta; \mathbf{W}) = \|\mathbf{y} - \sigma(\mathbf{W}\mathbf{x})\|^2$. Hence

$$g^{ij}(\eta; \mathbf{W}) = \nabla^{ij}_{\mathbf{W}} \mathcal{L}(\eta; \mathbf{W}) = -2(y_i - \sigma_i(\mathbf{W}\mathbf{x}))\sigma_i(\mathbf{W}\mathbf{x})(1 - \sigma_i(\mathbf{W}\mathbf{x}))x_j \tag{5.51}$$

Since $\mathbf{x} \in [0, 1]^{d_x}$, $\mathbf{y} \in [0, 1]^{d_y}$ and $\sigma(\cdot) \in [0, 1]$, we have

$$|g^{ij}(\eta; \mathbf{W})| = 2|y_i - \sigma_i(\mathbf{W}\mathbf{x})||\sigma_i(\mathbf{W}\mathbf{x})(1 - \sigma_i(\mathbf{W}\mathbf{x}))||x_j| \leqslant \frac{1}{2} \tag{5.52}$$

Hence $\mathbb{E}_\eta g^{ij}(\eta; \mathbf{W}) \leqslant \frac{1}{2}$. Using the definition of variance and (5.52) we then have

$$\begin{aligned} \mathrm{Var}_\eta(g^{ij}(\eta; \mathbf{W})) &= \mathbb{E}_\eta(g^{ij}(\eta; \mathbf{W}))^2 - (\mathbb{E}_\eta g^{ij}(\eta; \mathbf{W}))^2 \\ &\leqslant \mathbb{E}_\eta g^{ij}(\eta; \mathbf{W})(\frac{1}{2} - \mathbb{E}_\eta g^{ij}(\eta; \mathbf{W})) \leqslant \frac{1}{16} \end{aligned} \tag{5.53}$$

which follows from the fact that the global maxima of $t(c-t)$ is $\frac{c^2}{4}$. Using this in (5.50), we get

$$\mathbb{E}_\eta \|\delta\|^2 \leqslant \frac{d_x d_y}{16} \tag{5.54}$$

We now derive the second inequality required. Since $\nabla_{\mathbf{W}} f(\mathbf{W}) = \mathbb{E}_\eta \nabla_{\mathbf{W}} \mathcal{L}(\eta; \mathbf{W})$, we have

$$\begin{aligned} \|\nabla_{\mathbf{W}} f(\mathbf{W}) - \nabla_{\mathbf{W}} f(\hat{\mathbf{W}})\| &\leqslant \mathbb{E}_\eta \|\nabla_{\mathbf{W}} \mathcal{L}(\eta; \mathbf{W}) - \nabla_{\hat{\mathbf{W}}} \mathcal{L}(\eta; \hat{\mathbf{W}})\| \\ &\leqslant \sup_\eta \|\nabla_{\mathbf{W}} \mathcal{L}(\eta; \mathbf{W}) - \nabla_{\hat{\mathbf{W}}} \mathcal{L}(\eta; \hat{\mathbf{W}})\| \end{aligned} \tag{5.55}$$

Consider $|\nabla^{ij}_{\mathbf{W}} \mathcal{L}(\eta; \mathbf{W}) - \nabla^{ij}_{\mathbf{W}} \mathcal{L}(\eta; \hat{\mathbf{W}})|$ where $\nabla^{ij}_{\mathbf{W}} \mathcal{L}(\eta; \mathbf{W})$ is from (5.51). Adding subtracting $2(y_i - \sigma_i(\mathbf{W}\mathbf{x}))\sigma_i(\hat{\mathbf{W}}\mathbf{x})(1-\sigma_i(\hat{\mathbf{W}}\mathbf{x}))x_j$, and using triangle inequality we get

$$\begin{aligned} |\nabla^{ij}_{\mathbf{W}} \mathcal{L}(\eta; \mathbf{W}) - \nabla^{ij}_{\mathbf{W}} \mathcal{L}(\eta; \hat{\mathbf{W}})| \leqslant 2|&(y_i - \sigma_i(\mathbf{W}\mathbf{x}))\sigma_i(\mathbf{W}\mathbf{x})(1 - \sigma_i(\mathbf{W}\mathbf{x}))x_j \\ &- 2(y_i - \sigma_i(\mathbf{W}\mathbf{x}))\sigma_i(\hat{\mathbf{W}}\mathbf{x})(1 - \sigma_i(\hat{\mathbf{W}}\mathbf{x}))x_j| \\ &+ 2|(y_i - \sigma_i(\mathbf{W}\mathbf{x}))\sigma_i(\hat{\mathbf{W}}\mathbf{x})(1 - \sigma_i(\hat{\mathbf{W}}\mathbf{x}))x_j \\ &- (y_i - \sigma_i(\hat{\mathbf{W}}\mathbf{x}))\sigma_i(\hat{\mathbf{W}}\mathbf{x})(1 - \sigma_i(\hat{\mathbf{W}}\mathbf{x}))x_j| \end{aligned} \tag{5.56}$$

Rearranging the terms within each $|\cdot|$ we have

$$\begin{aligned} &|\nabla^{ij}_{\mathbf{W}} \mathcal{L}(\eta; \mathbf{W}) - \nabla^{ij}_{\mathbf{W}} \mathcal{L}(\eta; \hat{\mathbf{W}})| \\ &\quad\leqslant 2|x_j||y_i - \sigma_i(\mathbf{W}\mathbf{x})||\sigma_i(\mathbf{W}\mathbf{x})(1 - \sigma_i(\mathbf{W}\mathbf{x})) - \sigma_i(\hat{\mathbf{W}}\mathbf{x})(1 - \sigma_i(\hat{\mathbf{W}}\mathbf{x}))| \\ &\qquad + 2|\sigma_i(\mathbf{W}\mathbf{x}) - \sigma_i(\hat{\mathbf{W}}\mathbf{x})||\sigma_i(\mathbf{W}\mathbf{x})(1 - \sigma_i(\mathbf{W}\mathbf{x}))x_j| \end{aligned} \tag{5.57}$$

Using triangle inequality again and recalling that x_j, y_i and $\sigma_i(\cot)$ lie between 0 and 1, we have

$$
\begin{aligned}
&|\nabla^{ij}_{\mathbf{W}}\mathcal{L}(\eta;\mathbf{W}) - \nabla^{ij}_{\mathbf{W}}\mathcal{L}(\eta;\hat{\mathbf{W}})| \\
&\qquad \leqslant 2|\sigma_i(\mathbf{W}\mathbf{x}) - \sigma_i(\hat{\mathbf{W}}\mathbf{x})| + 2|\sigma_i^2(\mathbf{W}\mathbf{x}) - \sigma_i^2(\hat{\mathbf{W}}\mathbf{x})| \\
&\qquad\quad + 2|\sigma_i(\mathbf{W}\mathbf{x}) - \sigma_i(\hat{\mathbf{W}}\mathbf{x})||\sigma_i(\mathbf{W}\mathbf{x})(1-\sigma_i(\mathbf{W}\mathbf{x}))x_j| \\
&\qquad \leqslant 2|\sigma_i(\mathbf{W}\mathbf{x}) - \sigma_i(\hat{\mathbf{W}}\mathbf{x})| + 2|\sigma_i^2(\mathbf{W}\mathbf{x}) - \sigma_i^2(\hat{\mathbf{W}}\mathbf{x})| \\
&\qquad\quad + \frac{1}{2}|\sigma_i(\mathbf{W}\mathbf{x}) - \sigma_i(\hat{\mathbf{W}}\mathbf{x})|
\end{aligned}
\tag{5.58}
$$

where in the last step we used the fact that $\sigma_i(\hat{\mathbf{W}}\mathbf{x})(1-\sigma_i(\hat{\mathbf{W}}\mathbf{x})) \leqslant \frac{1}{4}$. Splitting the second term as $|\sigma_i^2(\mathbf{W}\mathbf{x}) - \sigma_i^2(\hat{\mathbf{W}}\mathbf{x})| = (\sigma_i(\mathbf{W}\mathbf{x}) + \sigma_i(\hat{\mathbf{W}}\mathbf{x}))|\sigma_i(\mathbf{W}\mathbf{x}) - \sigma_i(\hat{\mathbf{W}}\mathbf{x})|$ we have

$$
\begin{aligned}
|\nabla^{ij}_{\mathbf{W}}\mathcal{L}(\eta;\mathbf{W}) - \nabla^{ij}_{\mathbf{W}}\mathcal{L}(\eta;\hat{\mathbf{W}})| &\leqslant \frac{13}{2}|\sigma_i(\mathbf{W}\mathbf{x}) - \sigma_i(\hat{\mathbf{W}}\mathbf{x})| \\
&\leqslant \frac{13}{8}|\mathbf{W}_{i\cdot}\mathbf{x} - \hat{\mathbf{W}}_{i\cdot}\mathbf{x}| = \frac{13}{8}|\sum_j W_{ij}x_j - \sum_j \hat{W}_{ij}x_j| = \frac{13}{8}|W_{ij} - \hat{W}_{ij}|
\end{aligned}
\tag{5.59}
$$

where the second inequality above uses the Lipschitz constant of $\sigma(\cdot)$, and the third one uses Cauchy-Schwartz and the fact that $\mathbf{W}$ and $\hat{\mathbf{W}}$ differ only in ij^{th} entry.

Using this in (5.55) we have

$$
\begin{aligned}
\|\nabla_{\mathbf{W}} f(\mathbf{W}) - \nabla_{\mathbf{W}} f(\hat{\mathbf{W}})\| &\leqslant \sup_\eta \|\nabla_{\mathbf{W}}\mathcal{L}(\eta;\mathbf{W}) - \nabla_{\hat{\mathbf{W}}}\mathcal{L}(\eta;\hat{\mathbf{W}})\| \\
&= \sup_\eta \sqrt{\sum_{ij} |\nabla_{\mathbf{W}_{ij}}\mathcal{L}(\eta;\mathbf{W}) - \nabla_{\hat{\mathbf{W}}_{ij}}\mathcal{L}(\eta;\hat{\mathbf{W}})|} \\
&\leqslant \sqrt{\sum_{ij} (\frac{13}{8})^2 |W_{ij} - \hat{W}_{ij}|^2} = \frac{13}{8}\|\mathbf{W} - \hat{\mathbf{W}}\|
\end{aligned}
\tag{5.60}
$$

A significant part of the rest of the proof adapts that of Theorem 2.1 in Ghadimi and Lan (2013) with several adjustments. We first start with $f(\mathbf{W})$. Using (5.60), which corresponds to the Lipschitz constant for $f(\mathbf{W})$, we have

$$
f(\mathbf{W}^{k+1}) \leqslant f(\mathbf{W}^k) + \langle \nabla_{\mathbf{W}} f(\mathbf{W}^k), \mathbf{W}^{k+1} - \mathbf{W}^k \rangle + \frac{13}{16}\|\mathbf{W}^{k+1} - \mathbf{W}^k\|^2 \tag{5.61}
$$

where $\langle \cdot, \cdot \rangle$ denotes inner product. k denotes the iteration index $k = 1, \ldots, N$. Substituting for the update $\mathbf{W}^{k+1} \leftarrow \mathbf{W}^k - \gamma^k G(\eta^k; \mathbf{W}^k)$ (where γ^k is the stepsize) and using $G(\eta^k; \mathbf{W}^k) = \frac{1}{B}\sum_{b=1}^{B} g(\eta^{b,k}; \mathbf{W}^k)$, we get

$$f(\mathbf{W}^{k+1}) \leqslant f(\mathbf{W}^k) - \gamma^k \langle \nabla_{\mathbf{W}} f(\mathbf{W}^k), G(\eta^k; \mathbf{W}^k) \rangle + \frac{13}{16}(\gamma^k)^2 \|G(\eta^k; \mathbf{W}^k)\|^2$$
$$\leqslant f(\mathbf{W}^k) - \frac{\gamma^k}{B} \sum_{b=1}^{B} \langle \nabla_{\mathbf{W}} f(\mathbf{W}^k), g(\eta^{b,k}; \mathbf{W}^k) \rangle + \frac{13}{16B^2}(\gamma^k)^2 \| \sum_{b=1}^{B} g(\eta^{b,k}; \mathbf{W}^k)\|^2 \tag{5.62}$$

where $b = 1, \ldots, B$ represents indices within the mini-batch. $\eta^{b,k}$ denotes the $\{\mathbf{x}, \mathbf{y}\} \in \mathcal{X}$ sampled for b^{th} noisy gradient computation (within the mini-batch) at k^{th} iteration.
Let $\delta^{b,k} := g(\eta^{b,k}; \mathbf{W}^k) - \nabla_{\mathbf{W}} f(\mathbf{W}^k)$, and so we denote $\delta^k = \frac{1}{B} \sum_{b=1}^{B} \delta^{b,k} = G(\eta^k; \mathbf{W}) - \nabla_{\mathbf{W}} f(\mathbf{W})$. The above inequality then reduces to

$$f(\mathbf{W}^{k+1}) \leqslant f(\mathbf{W}^k) - \gamma^k \|\nabla_{\mathbf{W}} f(\mathbf{W}^k)\|^2 - \frac{\gamma^k}{B} \sum_{b=1}^{B} \langle \nabla_{\mathbf{W}} f(\mathbf{W}^k), \delta^{b,k} \rangle$$
$$+ \frac{13}{16B^2}(\gamma^k)^2 \|B \nabla_{\mathbf{W}} f(\mathbf{W}) + \sum_{b=1}^{B} \delta^{b,k}\|^2$$
$$f(\mathbf{W}^{k+1}) \leqslant f(\mathbf{W}^k) - \gamma^k \|\nabla_{\mathbf{W}} f(\mathbf{W}^k)\|^2 - \frac{\gamma^k}{B} \sum_{b=1}^{B} \langle \nabla_{\mathbf{W}} f(\mathbf{W}^k), \delta^{b,k} \rangle$$
$$+ \frac{13}{16B^2}(\gamma^k)^2 \left(B^2 \|\nabla_{\mathbf{W}} f(\mathbf{W})\|^2 + 2B \sum_{b=1}^{B} \langle \nabla_{\mathbf{W}} f(\mathbf{W}^k), \delta^{b,k} \rangle + \| \sum_{b=1}^{B} \delta^{b,k}\|^2 \right)$$
$$f(\mathbf{W}^{k+1}) \leqslant f(\mathbf{W}^k) - \left(\gamma^k - \frac{13}{16}(\gamma^k)^2 \right) \|\nabla_{\mathbf{W}} f(\mathbf{W}^k)\|^2$$
$$- \frac{1}{B} \left(\gamma^k - \frac{13}{8}(\gamma^k)^2 \right) \sum_{b=1}^{B} \langle \nabla_{\mathbf{W}} f(\mathbf{W}^k), \delta^{b,k} \rangle$$
$$+ \frac{13}{16B^2}(\gamma^k)^2 \| \sum_{b=1}^{B} \delta^{b,k}\|^2 \tag{5.63}$$

Adding up the above inequalities over the iterations $k = 1, \ldots, N$, we get

$$\sum_{k=1}^{N} \left(\gamma^k - \frac{13}{16}(\gamma^k)^2 \right) \|\nabla_{\mathbf{W}} f(\mathbf{W}^k)\|^2 \leqslant f(\mathbf{W}^1) - f(\mathbf{W}^{N+1})$$
$$- \sum_{k=1}^{N} \frac{1}{B} \left(\gamma^k - \frac{13}{8}(\gamma^k)^2 \right) \sum_{b=1}^{B} \langle \nabla_{\mathbf{W}} f(\mathbf{W}^k), \delta^{b,k} \rangle + \frac{13}{16B^2} \sum_{k=1}^{N} (\gamma^k)^2 \| \sum_{b=1}^{B} \delta^{b,k}\|^2 \tag{5.64}$$

$$\sum_{k=1}^{N}\left(\gamma^k-\frac{13}{16}(\gamma^k)^2\right)\|\nabla_{\mathbf{W}}f(\mathbf{W}^k)\|^2\leqslant f(\mathbf{W}^1)-f^*$$
$$-\frac{1}{B}\sum_{k=1}^{N}\sum_{b=1}^{B}\left(\gamma^k-\frac{13}{8}(\gamma^k)^2\right)\langle\nabla_{\mathbf{W}}f(\mathbf{W}^k),\delta^{b,k}\rangle+\frac{13}{16B^2}\sum_{k=1}^{N}(\gamma^k)^2\|\sum_{b=1}^{B}\delta^{b,k}\|^2 \tag{5.65}$$

where $\mathbf{W}^1$ is the initial estimate. The second inequality follows from the fact that at the optimum $\mathbf{W}^*$, we have $f^*\leqslant f(\mathbf{W}^{N+1})$.

Now observe that the above inequality comprises of two sets of random variables – sampling over η for constructing noisy gradients, and $N:=R\sim\mathbb{P}_R(\cdot)$ which denotes the stopping iteration (which is independent of η). We now marginalize out these variables from (5.65). Observe that, by definition, $\eta^{b,k}$ for different b's and k's are independent of each other. However, $\mathbf{W}^{k+1}$ depends on all $G(\eta^{b,l};\mathbf{W}^l)$s (being functions of $\eta^{b,k}$) for $l=1$ to k. Further, all the information needed for $k+1^{th}$ update comes from the "state" at k^{th} iteration, implying that the updates $\mathbf{W}^k$ form a *Markov* process. Hence we take the expectation with respect to $p(\eta^{[B,N]},R)=p(\eta^{[B,N]})p(R)$ where $\eta^{[B,N]}$ is the random process generating $\eta^{\cdot,1},\ldots,\eta^{\cdot,N}$.

Taking expectation over $\eta^{[B,N]}$, the second to last term of (5.65) becomes

$$\begin{aligned}\mathbb{E}_{\eta^{[B,N]}}&\left[\sum_{k=1}^{N}\sum_{b=1}^{B}\left(\gamma^k-\frac{13}{8}(\gamma^k)^2\right)\langle\nabla_{\mathbf{W}}f(\mathbf{W}^k),\delta^{b,k}\rangle\right]\\&=\sum_{k=1}^{N}\sum_{b=1}^{B}\left(\gamma^k-\frac{13}{8}(\gamma^k)^2\right)\ \mathbb{E}_{\eta^{[b,k]}}(\langle\nabla_{\mathbf{W}}f(\mathbf{W}^k),\delta^{b,k}\rangle|\eta^{1,1},\ldots,\eta^{b,k})\\&=0\end{aligned} \tag{5.66}$$

where the last equality follows from the definition of $\delta^{b,k}=g(\eta^{b,k};\mathbf{W}^k)-\nabla_{\mathbf{W}}f(\mathbf{W}^k)$ and the fact that $\mathbb{E}_\eta g(\eta;\mathbf{W}^k)=\nabla_{\mathbf{W}}f(\mathbf{W}^k)$.

To bound the last term in (5.65) consider the inner product $\langle\sum_{b=1}^{B-1}\delta^{b,k},\delta^{B,k}\rangle$. Taking the expectation with $\eta^{[B,k]}$ (for a given k), we have the following

$$\begin{aligned}&\mathbb{E}_{\eta^{[B,k]}}\left[\langle\sum_{b=1}^{B-1}\delta^{b,k},\delta^{B,k}\rangle|\eta^{1,k},\ldots,\eta^{B-1,k}\right]\\&=\langle\sum_{b=1}^{B-1}\delta^{b,k},\mathbb{E}_{\eta^{[B,k]}}\delta^{B,k}\rangle|\eta^{1,k},\ldots,\eta^{B-1,k}=0\end{aligned} \tag{5.67}$$

which follows from the fact that $\mathbb{E}_\eta\delta^{b,k}=\nabla_{\mathbf{W}}f(\mathbf{W})$. Using this we have the

following (the superscripts of $\eta^{\cdot}$ will be dropped hence forward so as to avoid reduce the clutter),

$$\begin{aligned}\mathbb{E}_\eta\|\sum_{b=1}^{B}\delta^{b,k}\|^2 &= \mathbb{E}_\eta\left[\|\sum_{b=1}^{B-1}\delta^{b,k}\|^2+\mathbb{E}_\eta\|\delta^{B,k}\|^2\right]+2\langle\sum_{b=1}^{B-1}\delta^{b,k},\delta^{B,k}\rangle\\ &= \mathbb{E}_\eta\|\sum_{b=1}^{B}\delta^{b,k}\|^2+\mathbb{E}_\eta\|\delta^{B,k}\|^2=\sum_{b=1}^{B}\mathbb{E}_\eta\|\delta^{b,k}\|^2\end{aligned} \tag{5.68}$$

Further, from (5.54) we have $\mathbb{E}_\eta\|\delta\|^2\leqslant\frac{d_x d_y}{16}$. Using this and (5.68), and taking the expectation over $\eta^{[B,N]}$ of the last term in (5.65), we get

$$\begin{aligned}&\mathbb{E}_\eta\left[\frac{13}{16B^2}\sum_{k=1}^{N}(\gamma^k)^2\|\sum_{b=1}^{B}\delta^{b,k}\|^2\right]\\ &\quad=\frac{13}{16B^2}\sum_{k=1}^{N}(\gamma^k)^2\mathbb{E}_\eta\|\sum_{b=1}^{B}\delta^{b,k}\|^2\leqslant\frac{13d_x d_y}{256B}\sum_{k=1}^{N}(\gamma^k)^2\end{aligned} \tag{5.69}$$

Using (5.66) and (5.69) and the inequality in (5.65) we have

$$\sum_{k=1}^{N}\left(\gamma^k-\frac{13}{16}(\gamma^k)^2\right)\mathbb{E}_\eta\|\nabla_{\mathbf{W}}f(\mathbf{W}^k)\|^2\leqslant f(\mathbf{W}^1)-f^*+\frac{e^s}{B}\sum_{k=1}^{N}(\gamma^k)^2 \tag{5.70}$$

where the definition e^s are used. Clearly for the above bound to make sense for any stepsize $\gamma^k<1$.

Recall that we are intersted in bounding the expected gradients $\mathbb{E}_{R,\eta}\|\nabla_{\mathbf{W}}f(\mathbf{W}^k)\|^2$. Apart from η, this expectation is over the stopping iteration R, where $R\sim\mathbb{P}_R(k)$ (i.e., sampled from the stopping distribution $\mathbb{P}_R(\cdot)$). Hence we intend to bound

$$\mathbb{E}_{R,\eta}\|\nabla_{\mathbf{W}}f(\mathbf{W}^k)\|^2:=\frac{1}{\sum_{k=1}^{N}p_R^k}\left(p_R^k\mathbb{E}_\eta\|\nabla_{\mathbf{W}}f(\mathbf{W}^k)\|^2\right) \tag{5.71}$$

where p_R^k is the probability of choosing k^{th} iteration to be the stopping iteration R (note that $k=1,\ldots,N$). Following the left hand side in (5.70), we see that p_R^k would depend on the stepsizes γ^k. Specifically, by choosing $p_R^k=\gamma^k-\frac{13}{16}(\gamma^k)^2$, we can compute the desired $\mathbb{E}_{R,\eta}\|\nabla_{\mathbf{W}}f(\mathbf{W}^k)\|^2$ as follows (clearly, whenever $\gamma^k<1$, $\gamma^k-\frac{13}{16}(\gamma^k)^2>0$, and hence it makes sense to use it as p_R^k),

$$\begin{aligned}\mathbb{E}_{R,\eta}\|\nabla_{\mathbf{W}}f(\mathbf{W}^k)\|^2&:=\frac{\sum_{k=1}^{N}\left(\gamma^k-\frac{13}{16}(\gamma^k)^2\right)\mathbb{E}_\eta\|\nabla_{\mathbf{W}}f(\mathbf{W}^k)\|^2}{\sum_{k=1}^{N}(\gamma^k-\frac{13}{16}(\gamma^k)^2)}\\ &\leqslant\frac{f(\mathbf{W}^1)-f^*+\frac{e^s}{B}\sum_{k=1}^{N}(\gamma^k)^2}{\sum_{k=1}^{N}(\gamma^k-\frac{13}{16}(\gamma^k)^2)}\end{aligned} \tag{5.72}$$

Observe that for the current result, however, we are interested in the constant stepsize case where $\gamma^k = \gamma \ \forall k$. Here the bounds, and the corresponding results, are a little more messier compared to the constant γ case, nevertheless, as will be shown in the proof of Corollary 5.3, the overall recipe for deriving the bound is the same even in the non-constant stepsize setting.
Using $\gamma^k = \gamma$ in (5.70) and (5.72) we get

$$\begin{aligned}\mathbb{E}_{R,\eta}\|\nabla_{\mathbf{W}} f(\mathbf{W}^k)\|^2 &:= \frac{1}{N\gamma e_\gamma^s}\left(f(\mathbf{W}^1) - f^* + \frac{e^s N\gamma^2}{B}\right) \\ &\leqslant \frac{1}{e_\gamma^s}\left(\frac{D_f}{N\gamma} + \frac{e^s\gamma}{B}\right)\end{aligned} \tag{5.73}$$

where $D_f = f(\mathbf{W}^1) - f^*$ is the initial deviation from optimum. Recall the discussion in the remarks of Theorem 5.1 (from Section 5.3), where we argue that using constant stepsizes with no other constraint on the stopping distribution is equivalent, from the perspective of the bound, to choosing the stopping distribution to be uniform over all the N iterations.
Observe that between the two terms within the parenthesis in (5.73), the first term decreases as γ increases, while the second term increases as γ increases. Balancing these two, we obtain the optimal constant stepsize γ_o (in the regime $\gamma < \frac{16}{13}$) as follows

$$\gamma_o = \sqrt{\frac{BD_f}{e^s N}} \approx \sqrt{\frac{Bf(\mathbf{W}^1)}{e^s N}} \tag{5.74}$$

where without loss of generality we used the approximation that $f^* \sim 0$ and hence $D_f \approx f(\mathbf{W}^1)$. Observe that instead of balancing these terms, we could have explicitly worked with the entire term in the bound and differentiated w.r.t γ, which will result in a quadratic involving γ. Note that unlike the above expression for γ_o, this latter case is much messier with no significant advantage.

Proof of Corollary 5.2

Using the result of Theorem 5.1 and substituting the optimal stepsize $\gamma_o = \sqrt{\frac{Bf(\mathbf{W}^1)}{e^s N}}$ in (5.73), with $e^s = \frac{13 d_x d_y}{256}$ and D_f replaced by $f(\mathbf{W}^1)$, we have

$$\mathbb{E}_{R,\eta}(\|\nabla_{\mathbf{W}} f(\mathbf{W}^R)\|^2) \leqslant \frac{d_x d_y \sqrt{13 f(\mathbf{W}^1)}}{8\sqrt{B}\left(\sqrt{d_x d_y N} - \sqrt{13 B f(\mathbf{W}^1)}\right)} \tag{5.75}$$

Observe that $\gamma < \frac{16}{13}$ ensures that $d_x d_y N > 13 B f(\mathbf{W}^1)$ thereby making

sure the denominator in the above inequality makes sense. We first derive the convergence rates of the above bound at this optimal choice of stepsize, followed by estimating the corresponding sample complexity using setup of (ϵ, δ)-solution which is based on choosing the best estimate from multiple runs of the Algorithm 2.

Although the right hand side above is non-trivial to interpret, the rates (vs. N and d_x, d_y) can be obtained as follows. For a given network (i.e., for a fixed $d_x d_y$) and batch size B, we have

$$\mathbb{E}_{R,\eta}(\|\nabla_{\mathbf{W}} f(\mathbf{W}^R)\|^2) \leqslant \frac{P}{\sqrt{N}} \quad \forall \quad N > N_o > \frac{13Bf(\mathbf{W}^1)}{d_x d_y}$$
$$\text{where} \quad P = \frac{a\sqrt{N_o}}{\sqrt{N_o} - b} \quad a = \sqrt{\frac{13 d_x d_y f(\mathbf{W}^1)}{64B}} \quad b = \sqrt{\frac{13Bf(\mathbf{W}^1)}{d_x d_y}} \tag{5.76}$$

$$\mathbb{E}_{R,\eta}(\|\nabla_{\mathbf{W}} f(\mathbf{W}^R)\|^2) \leqslant Q\sqrt{d_x d_y} \quad \forall \quad d_x d_y > d_o > \frac{13Bf(\mathbf{W}^1)}{N}$$
$$\text{where} \quad Q = \frac{\sqrt{d_o}}{a(\sqrt{d_o} - b)} \quad a = 8\sqrt{\frac{BN}{13f(\mathbf{W}^1)}} \quad b = \sqrt{\frac{13Bf(\mathbf{W}^1)}{N}} \tag{5.77}$$

Proof of Corollary 5.3

The proof for this result follows the same recipe as that of Theorem 5.1. We only point out the differences here.

In the non-constant stepsize case, using the stopping probability distribution such that $p_R^k = \gamma^k - \frac{13}{16}(\gamma^k)^2$ and (5.70), and computing the expected gradients both over R and η, we get the following

$$\mathbb{E}_{R,\eta}\|\nabla_{\mathbf{W}} f(\mathbf{W}^k)\|^2 \leqslant \frac{f(\mathbf{W}^1) - f^* + \frac{e^s \sum_{i=1}^N (\gamma^k)^2}{B}}{\sum_{k=1}^N (\gamma^k - \frac{13}{16}(\gamma^k)^2)} \tag{5.78}$$

Whenever $\gamma^k < 1$, we have $\gamma^k(1 - \frac{13\gamma^k}{16}) > \frac{3\gamma^k}{16}$, using which we have (and denoting $D_f = f(\mathbf{W}^1) - f^*$)

$$\mathbb{E}_{R,\eta}\|\nabla_{\mathbf{W}} f(\mathbf{W}^k)\|^2 \leqslant \frac{D_f + \frac{e^s \sum_{i=1}^N (\gamma^k)^2}{B}}{\frac{3}{16}\sum_{k=1}^N \gamma^k} = \frac{16}{3\sum_{k=1}^N \gamma^k}\left(D_f + \frac{e^s \sum_{k=1}^N (\gamma^k)^2}{B}\right) \tag{5.79}$$

Now using $\gamma^k = \frac{\gamma}{k^\alpha}$ for some $\alpha > 1$, we get

$$\begin{aligned}\mathbb{E}_{R,\eta}\|\nabla_{\mathbf{W}} f(\mathbf{W}^k)\|^2 &\leqslant \frac{16}{3\gamma\sum_{k=1}^{N} k^{-\alpha}}\left(D_f + \frac{e^s\gamma^2\sum_{k=1}^{N} k^{-2\alpha}}{B}\right)\\ &\leqslant \frac{16}{3\mathcal{H}_N(\alpha)}\left(\frac{D_f}{\gamma} + \frac{e^s\gamma\mathcal{H}_N(2\alpha)}{B}\right)\end{aligned} \tag{5.80}$$

where $\mathcal{H}_n(m) = \sum_{i=1}^{n}\frac{1}{i^m}(m > 1)$ is the generalized harmonic number. Comparing this to (5.73) we see that the change in the bound is with respect to some constants. Similar the computation for optimal stepsize as in (5.74), we have the following here (approximating D_f with $f(\mathbf{W}^1)$)

$$\gamma_o = \sqrt{\frac{Bf(\mathbf{W}^1)}{e^s\mathcal{H}_N(2\alpha)}} \tag{5.81}$$

Proof of Corollary 5.4

The proof for this result follows the same recipe as that of Theorem 5.1. We only point out the differences here.
Recall (5.70) from the proof of Theorem 5.1 earlier. Using constant stepsizes $\gamma^k = \gamma$ and the definitions of e^s and e^s_γ, this would reduce to

$$\sum_{k=1}^{N}\mathbb{E}_\eta\|\nabla_{\mathbf{W}} f(\mathbf{W}^k)\|^2 \leqslant \frac{1}{\gamma e^s_\gamma}\left(f(\mathbf{W}^1) - f^* + \frac{e^s N}{B}\gamma^2\right) \tag{5.82}$$

Following a similar recipe as that of (5.71)-(5.73), we have the following, where, without the loss of generality, we assume that the probabilities p_R^k are normalized i.e., $\sum_{k=1}^{N} p_R^k = 1$ and $p_R^k \leqslant 1$,

$$\mathbb{E}_{R,\eta}\|\nabla_{\mathbf{W}} f(\mathbf{W}^k)\|^2 := \sum_{k=1}^{N}\left(p_R^k\mathbb{E}_\eta\|\nabla_{\mathbf{W}} f(\mathbf{W}^k)\|^2\right) \tag{5.83}$$

and using the condition that $p_R^k \leqslant p_R^{k+1}$ for all $k = 2,\ldots,N$ and replacing $f(\mathbf{W}^1) - f^*$ with D_f, we have

$$\mathbb{E}_{R,\eta}\|\nabla_{\mathbf{W}} f(\mathbf{W}^k)\|^2 \leqslant \frac{p_R^N}{\gamma e^s_\gamma}\left(D_f + \frac{e^s N}{B}\gamma^2\right) \tag{5.84}$$

Depending on p_R^N, the right hand side above can be very lose, and hence it does not necessarily make sense to compute the optimal stepsize for any general $\mathbb{P}_R(\cdot)$. Nevertheless, following the optimal stepsize computation above in (5.74), we have (using $D_f \approx f(\mathbf{W}^1)$)

$$\gamma_o^1 = \gamma = \sqrt{\frac{Bf(\mathbf{W}^1)}{e^s N}} \tag{5.85}$$

Proof of Theorem 5.6

Recall that, for a given $\epsilon > 0$ and $0 < \delta \ll 1$), the (ϵ, δ)-solution is defined as follows:

$$\mathbf{W}^* = \arg\min_t \|\nabla_{\mathbf{W}} f(\mathbf{W}^{R_t})\|^2 \quad \text{s.t.} \quad Pr\left(\min_t \|\nabla_{\mathbf{W}} f(\mathbf{W}^{R_t})\|^2 \leqslant \epsilon\right) \geqslant 1 - \delta \tag{5.86}$$

where $\mathbf{W}^{R_1}, \ldots, \mathbf{W}^{R_T}$ correspond to estimates from the T independent runs of Algorithm 2. Without loss of generality we assume that N and B are the same across all these T runs.
Using some basic probability properties,

$$\begin{aligned} Pr\left(\min_t \|\nabla_{\mathbf{W}} f(\mathbf{W}^{R_t})\|^2 \leqslant \epsilon\right) &= Pr\left(\exists t = 1, \ldots, T \text{ s.t. } \|\nabla_{\mathbf{W}} f(\mathbf{W}^{R_t})\|^2 \leqslant \epsilon\right) \\ &= 1 - Pr\left(\|\nabla_{\mathbf{W}} f(\mathbf{W}^{R_t})\|^2 \geqslant \epsilon \ \forall \ t = 1, \ldots, T\right) \\ &= 1 - \prod_{t=1}^{T} Pr\left(\|\nabla_{\mathbf{W}} f(\mathbf{W}^{R_t})\|^2 \geqslant \epsilon\right) \end{aligned} \tag{5.87}$$

Observe that we want the above probability to be larger than a given $1 - \delta$. Hence we require

$$\prod_{t=1}^{T} Pr\left(\|\nabla_{\mathbf{W}} f(\mathbf{W}^{R_t})\|^2 \geqslant \epsilon\right) \leqslant \delta \tag{5.88}$$

Now recall the constraint on the stepsizes and stopping distribution i.e., $\gamma^k = \gamma$, $p_R^k \leqslant p_R^{k+1} \forall k$, $p_R^N = \frac{\vartheta}{N}$. Using this and the optimal value for γ from (5.85) with $D_f \approx f(\mathbf{W}^1)$, and substituting it in (5.84), we get

$$\begin{aligned} &\frac{p_R^N}{\gamma_o(1 - \frac{13}{16}\gamma_o)}\left(D_f + \frac{1}{B}e^s N \gamma_o^2\right) \quad \text{with} \quad \gamma_o = \sqrt{\frac{Bf(\mathbf{W}^1)}{e^s N}} \\ &\qquad\qquad \text{gives} \quad \frac{32 p_R^N e^s N \sqrt{f(\mathbf{W}^1)}}{\sqrt{B}(16\sqrt{e^s N} - 13\sqrt{Bf(\mathbf{W}^1)})} \end{aligned} \tag{5.89}$$

where the definition of $e_\gamma^s = 1 - \frac{13\gamma}{16}$ was used.
Now using Markov inequality and optimal stepsize reduced bound from (5.89), we have the following

$$\Pr\left(\|\nabla_{\mathbf{W}} f(\mathbf{W}^{R_t})\|^2 \geqslant \epsilon\right) \leqslant \frac{\mathbb{E}(\nabla_{\mathbf{W}} f(\mathbf{W}^{R_t})\|^2)}{\epsilon} \leqslant \frac{32e^s\vartheta\sqrt{f(\mathbf{W}^1)}}{\epsilon\sqrt{B}(16\sqrt{e^s N} - 13\sqrt{Bf(\mathbf{W}^1)})} \tag{5.90}$$

Noting that the T different runs of Algorithm 2 are independent, we can ensure (5.88) as follows

$$\begin{aligned} &\left(\frac{32e^s\sqrt{f(\mathbf{W}^1)}}{\epsilon\sqrt{B}(16\sqrt{e^s N} - 13\sqrt{Bf(\mathbf{W}^1)})}\right)^T \leqslant \delta \\ &\iff 16\sqrt{e^s N} - 13\sqrt{Bf(\mathbf{W}^1)} \geqslant \frac{32e^s\vartheta\sqrt{f(\mathbf{W}^1)}}{\epsilon\delta^{1/T}\sqrt{B}} \\ &\iff N \geqslant \frac{f(\mathbf{W}^1)}{256e^s}\left(13\sqrt{B} + \frac{32e^s\vartheta}{\epsilon\delta^{1/T}\sqrt{B}}\right)^2 \end{aligned} \tag{5.91}$$

Using $\bar{\delta} = \frac{13B\epsilon\delta^{1/T}}{32e^s}$, and observing that the denominator in (5.89) is positive for any stepsize $\gamma < 1$ (because for such stepsizes we will have $(1 - \frac{13}{16}\gamma) > 0$), we finally have

$$N \geqslant \frac{4f(\mathbf{W}^1)e^s}{B\delta^{2/T}\epsilon^2}(\bar{\delta} + \vartheta)^2 \tag{5.92}$$

Proof of Corollary 5.7

Inducing dropout into the setting of 1-NN from Theorem 5.1 corresponds to adjusting the terms in (5.50)–(5.73).

To see this, first observe that dropout corresponds to dropping out units, and we consider the setting where this dropping is in the inputs/visible layer, and there is no dropout in the output layer. Specifically, in each iteration of Algorithm 2, an input feature/dimension is dropped with probability $1 - \zeta$. Hence, within each iteration, out of the $d_x d_y$ number of unknowns, only $\zeta d_x d_y$ are updated. If $\mathcal{J}$ denotes the ζd_x *non-dropped* dimensions, then within a given iteration, the following statements will hold.

$$\nabla^{ij}_W f(\mathbf{W}) = 0 \quad ; \quad g^{ij}(\eta; \mathbf{W}) = \nabla^{ij}_{\mathbf{W}} \mathcal{L}(\eta; \mathbf{W}) = 0 \qquad j \in \mathcal{J}^C \tag{5.93}$$

The first and second equalities are due to the fact that, within an iteration we have a network of ζd_x and d_y outputs, and the transformation matrix will be of size $d_y \times \zeta d_x$. These imply that the variance bound in (5.54) would change from $\frac{d_x d_y}{16}$ to $\frac{\zeta d_x d_y}{16}$.

Further, the inequalities from (5.61)–(5.63) and (5.64)–(5.65) would have to be *corrected* for this change in the number of unknowns being updated within each iteration. This can be done by observing changing the left hand side of (5.70). Note that the $\zeta d_x d_y$ unknowns across different iterations are independent, and a given edge is dropped w.p. $1 - zeta$. Hence, whenever N is reasonably large, the effective number of times a given W_{ij} is updated is ζN i.e., a given W_{ij} appears ζN times in the left hand side of (5.70) (instead of N as in the non-dropout setting).

Overall, this amounts to the following changes. First, the constants e^s would need to be scaled down to ζe^s (recall the definition of e^s here). Second, the left hand side in the inequality from (5.70) would change to

$$\zeta \sum_{k=1}^{N} \left(\gamma^k - \frac{13}{16} (\gamma^k)^2 \right) \mathbb{E}_\eta \|\nabla_{\mathbf{W}} f(\mathbf{W}^k)\|^2 \tag{5.94}$$

Incorporating these into (5.70)–(5.73), we have

$$\begin{aligned} \mathbb{E}_{R,\eta} \|\nabla_{\mathbf{W}} f(\mathbf{W}^k)\|^2 &:= \frac{1}{\zeta N \gamma e_\gamma^s} \left(f(\mathbf{W}^1) - f^* + \frac{\zeta e^s N \gamma^2}{B} \right) \\ &\leqslant \frac{1}{e_\gamma^s} \left(\frac{D_f}{N \gamma \zeta} + \frac{e^s \gamma}{B} \right) \end{aligned} \tag{5.95}$$

where $D_f = f(\mathbf{W}^1) - f^*$ is the initial deviation from optimum. The corresponding optimal stepsize would simply be

$$\gamma_o = \sqrt{\frac{B D_f}{\zeta e^s N}} \approx \sqrt{\frac{B f(\mathbf{W}^1)}{\zeta e^s N}} \tag{5.96}$$

To compute the computational complexity, following the steps from the proof of Corollary 5.6 from Section 5.9, we have the following. Using (5.86)–(5.88), and (5.95) and (5.96) from above, we have

$$\begin{aligned} \frac{1}{(1 - \frac{13}{16}\gamma_o)} \left(\frac{D_f}{N \gamma_o \zeta} + \frac{e^s \gamma_o}{B} \right) \quad \text{with} \quad \gamma_o = \sqrt{\frac{B f(\mathbf{W}^1)}{\zeta e^s N}} \\ \text{gives} \quad \frac{32 e^s \sqrt{f(\mathbf{W}^1)}}{\sqrt{B}(16\sqrt{\zeta e^s N} - 13\sqrt{B f(\mathbf{W}^1)})} \end{aligned} \tag{5.97}$$

Using this, the definition of $\bar{\delta}$ and following the steps from (5.90)–(5.92), we get

$$N \geqslant \frac{4 f(\mathbf{W}^1) e^s}{\zeta B \delta^{2/T} \epsilon^2} (1 + \bar{\delta})^2 \tag{5.98}$$

Proof of Theorem 5.8

This proof structure closely follows that of Theorem 5.1. Following the setup there, we first derive the inequality concerning the behaviour of the noisy gradients (refer to (5.54)). Using this, we then bound the expected gradients taking into account that, unlike the single-layer case, the minimization is over box constrained set $W_{ij} \in [-w_m, w_m]$ ($\forall i, j$; refer to Algorithm 3). The update step in Algorithm 3 comprises of projecting the parameter update onto this feasible set.

The loss function for DA pretraining is $\mathcal{L}(\eta; \mathbf{W}) = \|\mathbf{x} - \sigma(\mathbf{W}^T \sigma(\mathbf{W}\tilde{\mathbf{x}}))\|^2$ (recall the objective $f(\mathbf{W}) = \mathbb{E}_\eta \mathcal{L}(\eta; \mathbf{W})$). The minimization is over $W_{ij} \leqslant w_m$ for $i = 1, \ldots, d_h$ and $j = 1, \ldots, d_x$ with $\tilde{x}_j = x_j$ w.p ζ and 0 otherwise. For some $I = 1, \ldots, d_h$ and $J = 1, \ldots, d_x$, consider the gradient of this loss with respect to W_{IJ}, given as follows

$$\begin{aligned} g^{IJ}(\eta; \mathbf{W}) &= \nabla^{IJ}_{\mathbf{W}} \mathcal{L}(\eta; \mathbf{W}) \\ &= \sum_{j=1}^{d_x} -2(x_j - \sigma_j(\mathbf{W}^T \sigma(\mathbf{W}\tilde{\mathbf{x}})))\sigma_j(\mathbf{W}^T \sigma(\mathbf{W}\tilde{\mathbf{x}}))(1 - \sigma_j(\mathbf{W}^T \sigma(\mathbf{W}\tilde{\mathbf{x}}))) \\ &\qquad [\mathbb{I}_J \sigma_I(\mathbf{W}\tilde{\mathbf{x}}) + W_{Ij}\sigma_I(\mathbf{W}\tilde{\mathbf{x}})(1 - \sigma_I(\mathbf{W}\tilde{\mathbf{x}}))\tilde{x}_J] \end{aligned} \tag{5.99}$$

where $\mathbb{I}_J = 1$ for $j = J$ and 0 else.

Separating the cases of $\tilde{x}_J = 0$ and $\tilde{x}_J = x_J$, and using the properties of $\sigma(\cdot)$, and the fact that $\mathbf{x} \in [0, 1]^{d_x}$ we have the following

$$
\begin{aligned}
&|g^{IJ}(\eta;\mathbf{W})| \quad |\{\tilde{x}_J = 0\} \\
&= |2(x_J - \sigma_J(\mathbf{W}^T\sigma(\mathbf{W}\tilde{\mathbf{x}})))\sigma_J(\mathbf{W}^T\sigma(\mathbf{W}\tilde{\mathbf{x}}))(1-\sigma_J(\mathbf{W}^T\sigma(\mathbf{W}\tilde{\mathbf{x}})))\sigma_I(\mathbf{W}\tilde{\mathbf{x}})| \leqslant \frac{1}{2} \\
&|g^{IJ}(\eta;\mathbf{W})| \quad |\{\tilde{x}_J = x_J\} \\
&\quad = \quad |2(x_J - \sigma_J(\mathbf{W}^T\sigma(\mathbf{W}\tilde{\mathbf{x}})))\sigma_J(\mathbf{W}^T\sigma(\mathbf{W}\tilde{\mathbf{x}}))(1-\sigma_J(\mathbf{W}^T\sigma(\mathbf{W}\tilde{\mathbf{x}})))) \\
&\qquad (\sigma_I(\mathbf{W}\tilde{\mathbf{x}}) + W_{IJ}\sigma_I(\mathbf{W}\tilde{\mathbf{x}})(1-\sigma_I(\mathbf{W}\tilde{\mathbf{x}}))x_J) \\
&\qquad + 2\sum_{j\neq J}(x_j - \sigma_j(\mathbf{W}^T\sigma(\mathbf{W}\tilde{\mathbf{x}})))\sigma_j(\mathbf{W}^T\sigma(\mathbf{W}\tilde{\mathbf{x}}))(1-\sigma_j(\mathbf{W}^T\sigma(\mathbf{W}\tilde{\mathbf{x}}))) \\
&\qquad\qquad W_{Ij}\sigma_I(\mathbf{W}\tilde{\mathbf{x}})(1-\sigma_I(\mathbf{W}\tilde{\mathbf{x}})\tilde{x}_J| \\
&\quad \leqslant \quad \frac{1}{2}|x_J - \sigma_J(\mathbf{W}^T\sigma(\mathbf{W}\tilde{\mathbf{x}}))||\sigma_I(\mathbf{W}\tilde{\mathbf{x}}) + W_{IJ}\sigma_I(\mathbf{W}\tilde{\mathbf{x}})(1-\sigma_I(\mathbf{W}\tilde{\mathbf{x}}))x_J| \\
&\qquad + \frac{1}{2}x_J\sum_{j\neq J}|x_j - \sigma_j(\mathbf{W}^T\sigma(\mathbf{W}\tilde{\mathbf{x}}))|\frac{1}{4}|W_{Ij}| \\
&\quad \leqslant \quad \frac{1}{2}|x_J - \sigma_J(\mathbf{W}^T\sigma(\mathbf{W}\tilde{\mathbf{x}}))| + |x_J - \sigma_J(\mathbf{W}^T\sigma(\mathbf{W}\tilde{\mathbf{x}}))||W_{IJ}|\frac{1}{4}x_J \\
&\qquad + \frac{1}{2}x_J\sum_{j\neq J}|x_j - \sigma_j(\mathbf{W}^T\sigma(\mathbf{W}\tilde{\mathbf{x}}))|\frac{1}{4}|W_{Ij}| \quad \leqslant \quad \frac{1}{2} + \frac{x_J}{8}\sum_{j=1}^{d_x}|W_{Ij}|
\end{aligned}
\tag{5.100}
$$

Taking expectation over $\mathbf{x} \in \mathcal{X}$ for a given corruption of J^{th} unit (i.e., independent of the input $\mathbf{x}$, $\tilde{x}_J$ is always 0), we get

$$
\mathbb{E}_{\mathbf{x}}(g^{IJ}(\eta;\mathbf{W})) \leqslant \begin{cases} \frac{1}{2} & \text{whenever } \tilde{x}_J = 0 \\ \frac{1}{2} + \frac{\mu_J d_x w_m}{8} & \text{otherwise} \end{cases} \tag{5.101}
$$

where the second case is because $\mathbb{E}_{\mathbf{x}}(x_J\sum_{j=1}^{d_x}|W_{Ij}|) \leqslant \mathbb{E}_{\mathbf{x}}(x_J d_x w_m) = \mu_J d_x w_m$ which uses the max-norm boundary (or box) constraint for the parameters i.e., $W_{ij} \leqslant w_m \forall i,j$ (refer to the update from Algorithm 3). Here $\mu_j = \mathbb{E}_{\mathbf{x}}x_j$.
Further, using the definition of variance we have

$$
\begin{aligned}
\mathrm{Var}_{\mathbf{x}}(g^{IJ}(\eta;\mathbf{W})) &\leqslant \mathbb{E}_{\mathbf{x}}(g^{IJ}(\eta;\mathbf{W}))\left(sup_{\mathbf{x}}(g^{IJ}(\eta;\mathbf{W})) - \mathbb{E}_{\mathbf{x}}(g^{IJ}(\eta;\mathbf{W}))\right) \\
&\leqslant \begin{cases} \frac{1}{16} & \text{whenever } \tilde{x}_J = 0 \\ \frac{1}{16}\left(1 + \frac{\mu_J d_x w_m}{4}\right)^2 & \text{otherwise} \end{cases}
\end{aligned}
\tag{5.102}
$$

Note that we are interested in $\mathrm{Var}_{\eta}(G^{ij}(\eta;\mathbf{W}))$ for some i,j. Here $\mu_j = \mathbb{E}_{\mathbf{x}}x_j$ is the average (across all inputs from $\mathcal{X}$) first moment at the j^{th} input unit. Hence using the variance decomposition, we get

$$\begin{aligned}\mathrm{Var}_{\eta}(g^{ij}(\eta;\mathbf{W})) &= \mathrm{Var}_{\mathbf{x},\tilde{\mathbf{x}}}(g^{ij}(\eta;\mathbf{W}))\\ &= \mathbb{E}_{\tilde{\mathbf{x}}|\mathbf{x}}[\mathrm{Var}_{\mathbf{x}}(g^{ij}(\eta;\mathbf{W}))|\{\tilde{\mathbf{x}}|\mathbf{x}\}] + \mathrm{Var}_{\tilde{\mathbf{x}}|\mathbf{x}}[\mathbb{E}_{\mathbf{x}}(g^{ij}(\eta;\mathbf{W}))|\{\tilde{\mathbf{x}}|\mathbf{x}\}]\\ &= \mathbb{E}_{\tilde{x}_j|x_j}[\mathrm{Var}_{\mathbf{x}}(g^{ij}(\eta;\mathbf{W}))|\{\tilde{x}_j|x_j\}] + \mathrm{Var}_{\tilde{x}_j|x_j}[\mathbb{E}_{\mathbf{x}}(g^{ij}(\eta;\mathbf{W}))|\{\tilde{x}_j|x_j\}]\end{aligned} \tag{5.103}$$

Now applying (5.102) and (5.103) and using the fact that $\tilde{x}_j = 0$ w.p $1-\zeta$, we have

$$\mathbb{E}_{\tilde{x}_j|x_j}[\mathrm{Var}_{\mathbf{x}}(g^{ij}(\eta;\mathbf{W}))|\{\tilde{x}_j|x_j\}] \leqslant \frac{1-\zeta}{16} + \frac{\zeta}{16}\left(1+\frac{\mu_j d_x w_m}{4}\right)^2 \tag{5.104}$$

$$\begin{aligned}&\mathrm{Var}_{\tilde{x}_j|x_j}[\mathbb{E}_{\mathbf{x}}(g^{ij}(\eta;\mathbf{W}))|\{\tilde{x}_j|x_j\}]\\ &\quad\leqslant \mathbb{E}_{\tilde{x}_j|x_j}[\mathbb{E}^2_{\mathbf{x}}(G^{ij}(\eta;\mathbf{W}))] - (\mathbb{E}_{\tilde{x}_j|x_j}[\mathbb{E}_{\mathbf{x}}(g^{ij}(\eta;\mathbf{W}))])^2\\ &\quad\leqslant \frac{1-\zeta}{4} + \zeta\left(\frac{1}{2}+\frac{\mu_j d_x w_m}{8}\right)^2 - \left(\frac{1-\zeta}{2}+\frac{\zeta}{2}+\frac{\zeta\mu_j d_x w_m}{8}\right)^2\end{aligned} \tag{5.105}$$

Combining these we finally get

$$\mathrm{Var}_{\eta}(g^{ij}(\eta;\mathbf{W})) \leqslant \frac{1}{16}\left[1+\frac{\zeta\mu_j d_x w_m}{4} + \left(\frac{5\zeta}{16}-\frac{\zeta^2}{4}\right)(\zeta\mu_j d_x w_m)^2\right] \tag{5.106}$$

Following the setup from (5.50), (5.53) and (5.54) from the proof of Theorem 5.1 in Section 5.9, and replicating the same here, we have

$$\begin{aligned}\mathbb{E}_{\eta}\|\delta\|^2 &\leqslant \frac{1}{16}\left[d_x d_h + \frac{\zeta d_x w_m}{4}\sum_{ij}\mu_j + \left(\frac{5\zeta}{16}-\frac{\zeta^2}{4}\right)(\zeta d_x w_m)^2\sum_{ij}\mu_j^2\right]\\ &= \frac{1}{16}\left[d_x d_h + \frac{\zeta d_x d_h w_m}{4}\sum_{j}\mu_j + \left(\frac{5\zeta}{16}-\frac{\zeta^2}{4}\right)(\zeta d_x w_m)^2 d_h\sum_{j}\mu_j^2\right]\\ &= \frac{d_x d_h}{16}\left[1+\frac{\zeta d_x w_m}{4}\mu_{\mathbf{x}} + \left(\frac{5\zeta}{16}-\frac{\zeta^2}{4}\right)(\zeta d_x w_m)^2\tau_{\mathbf{x}}\right]\end{aligned} \tag{5.107}$$

where $\mu_{\mathbf{x}} = \frac{1}{d_x}\sum_{j=1}^{d_x}\mu_j$ and $\tau_{\mathbf{x}} = \frac{1}{d_x}\sum_{j=1}^{d_x}\mu_j^2$ correspond to the average moment of the inputs across all d_x units/dimensions. Note that $\tau_{\mathbf{x}}$ is not the second moment, and instead is a function of the first moments (μ_j's) themselves. Recall that $\delta = g(\eta;\mathbf{W}) - \nabla_{\mathbf{W}} f(\mathbf{W})$.

The rest of the proof follows the recipe from Section 5.9 and builds the expected bounds for Algorithm 3 following the update

$$\mathbf{W}_{ij}^{k+1} \leftarrow P_{\mathbf{W}}(\mathbf{W}_{ij}^{k} - \gamma^k G(\eta^k;\mathbf{W}_{ij}^{k})) \tag{5.108}$$

Observe that $P_{\mathbf{W}}(\cdot)$ denotes the Euclidean projection on $[-w_m, w_m]$. We will

be needing the following results about the Euclidean projection for constructing the convergence bound.

$$\begin{aligned} &\text{(a)} \quad h^{\mathsf{T}} P_{\mathbf{W}}(\mathbf{W}, h, \gamma) \geqslant \|P_{\mathbf{W}}(\mathbf{W}, h, \gamma)\|^2 \\ &\text{(b)} \quad \|P_{\mathbf{W}}(\mathbf{W}, h_1, \gamma) - P_{\mathbf{W}}(\mathbf{W}, h_2, \gamma)\| \leqslant \|h_1 - h_2\| \end{aligned} \tag{5.109}$$

for some h, h_1, $h_2 \in \mathbb{R}^{d_h d_x}$. We do not discuss their proofs here, which can be found in Rockafellar and Wets (2009) (and Lemma 1 and Proposition 2 in Ghadimi et al. (2016)). (5.109)(b) corresponds to the general inequality that projection does not reduce distances. For simplicity we have the following notation for the projected versions of $g(\eta; \mathbf{W})$, $G(\eta; \mathbf{W})$ and $\nabla_{\mathbf{W}} f(\eta; \mathbf{W})$.

$$\begin{aligned} P_{\mathbf{W}}(\mathbf{W}, g(\eta; \mathbf{W}), \gamma) &:= \tilde{g}(\eta; \mathbf{W}) \\ P_{\mathbf{W}}(\mathbf{W}, G(\eta; \mathbf{W}), \gamma) &:= \tilde{G}(\eta; \mathbf{W}) \\ P_{\mathbf{W}}(\mathbf{W}, \nabla_{\mathbf{W}} f(\mathbf{W}), \gamma) &:= \nabla_{\mathbf{W}} \tilde{f}(\mathbf{W}) \end{aligned} \tag{5.110}$$

(i.e, by adding an additional *tilde*).
We now follow the recipe from (5.61)–(5.73), from the proof of Theorem 5.1. Using the Lipschitz constant $\mathcal{U}_{da}$ (and denoting it by $L(d_h, w_m)$), we have the following

$$\begin{aligned} f(\mathbf{W}^{k+1}) &\leqslant f(\mathbf{W}^k) + \langle \nabla_{\mathbf{W}} f(\mathbf{W}^k), \mathbf{W}^{k+1} - \mathbf{W}^k \rangle + \frac{\mathcal{U}_{da}}{2} \|\mathbf{W}^{k+1} - \mathbf{W}^k\|^2 \\ \text{where} \quad \mathcal{U}_{da} &= \left((d_h - 1) \left[\frac{w_m}{20} + \frac{13 w_m^2}{640} \right] + \frac{16}{20} + \frac{73 w_m}{480} \right) \end{aligned} \tag{5.111}$$

By the definition of prox operation, and using some abuse of notation, we have $\mathbf{W}^{k+1} \leftarrow \mathbf{W}^k - \gamma^k \tilde{G}(\eta^k; \mathbf{W}^k)$, and recall that $\delta^k = G(\eta^k; \mathbf{W}^k) - \nabla_{\mathbf{W}} f(\mathbf{W}^k)$. Using these we get

$$\begin{aligned} f(\mathbf{W}^{k+1}) &\leqslant f(\mathbf{W}^k) + \langle \nabla_{\mathbf{W}} f(\mathbf{W}^k), \mathbf{W}^{k+1} - \mathbf{W}^k \rangle + \frac{\mathcal{U}_{da}}{2} \|\mathbf{W}^{k+1} - \mathbf{W}^k\|^2 \\ &= f(\mathbf{W}^k) - \gamma^k \langle \nabla_{\mathbf{W}} f(\mathbf{W}^k), \tilde{G}(\eta^k; \mathbf{W}^k) \rangle + \frac{\mathcal{U}_{da}}{2} (\gamma^k)^2 \|\tilde{G}(\eta^k; \mathbf{W}^k)\|^2 \\ &= f(\mathbf{W}^k) - \gamma^k \langle G(\eta^k; \mathbf{W}^k), \tilde{G}(\eta^k; \mathbf{W}^k) \rangle + \frac{\mathcal{U}_{da}}{2} (\gamma^k)^2 \|\tilde{G}(\eta^k; \mathbf{W}^k)\|^2 \\ &\qquad + \gamma^k \langle \delta^k, \tilde{G}(\eta^k; \mathbf{W}^k) \rangle \end{aligned} \tag{5.112}$$

and using (5.109)(a) with $h = G(\eta^k; \mathbf{W}^k)$, we get

$$
\begin{aligned}
f(\mathbf{W}^{k+1}) &\leqslant f(\mathbf{W}^k) - \gamma^k \|\tilde{G}(\eta^k; \mathbf{W}^k)\|^2 + \frac{\mathcal{U}_{da}}{2}(\gamma^k)^2 \|\tilde{G}(\eta^k; \mathbf{W}^k)\|^2 \\
&\quad + \gamma^k \langle \delta^k, \tilde{G}(\eta^k; \mathbf{W}^k) \rangle \\
&= f(\mathbf{W}^k) - \gamma^k \|\tilde{G}(\eta^k; \mathbf{W}^k)\|^2 + \frac{\mathcal{U}_{da}}{2}(\gamma^k)^2 \|\tilde{G}(\eta^k; \mathbf{W}^k)\|^2 \\
&\quad + \gamma^k \langle \delta^k, \nabla_{\mathbf{W}} \tilde{f}(\eta^k; \mathbf{W}^k) \rangle + \gamma^k \langle \delta^k, \tilde{G}(\eta^k; \mathbf{W}^k) - \nabla_{\mathbf{W}} \tilde{f}(\mathbf{W}^k) \rangle \\
&= f(\mathbf{W}^k) - \left(\gamma^k - \frac{\mathcal{U}_{da}}{2}(\gamma^k)^2 \right) \|\tilde{G}(\eta^k; \mathbf{W}^k)\|^2 \\
&\quad + \gamma^k \langle \delta^k, \nabla_{\mathbf{W}} \tilde{f}(\eta^k; \mathbf{W}^k) \rangle + \gamma^k \|\delta^k\| \|\tilde{G}(\eta^k; \mathbf{W}^k) - \nabla_{\mathbf{W}} \tilde{f}(\mathbf{W}^k)\|
\end{aligned}
\tag{5.113}
$$

where the last term in the third inequality above is obtained using Cauchy-Schwartz. Using (5.109)(b) with $h_1 = G(\eta^k; \mathbf{W}^k)$ and $h_2 = \nabla_{\mathbf{W}} f(\eta; \mathbf{W})$ in the last inequality above, we have

$$
\begin{aligned}
f(\mathbf{W}^{k+1}) \leqslant f(\mathbf{W}^k) &- \left(\gamma^k - \frac{\mathcal{U}_{da}}{2}(\gamma^k)^2 \right) \|\tilde{G}(\eta^k; \mathbf{W}^k)\|^2 \\
&+ \gamma^k \langle \delta^k, \nabla_{\mathbf{W}} \tilde{f}(\mathbf{W}^k) \rangle + \gamma^k \|\delta^k\|^2
\end{aligned}
\tag{5.114}
$$

Adding up the above inequalities over the iterations $k = 1, \ldots, N$, we get

$$
\begin{aligned}
\sum_{k=1}^{N} \left(\gamma^k - \frac{\mathcal{U}_{da}}{2}(\gamma^k)^2 \right) \|\tilde{G}(\eta^k; \mathbf{W}^k)\|^2 &\leqslant f(\mathbf{W}^1) - f(\mathbf{W}^{N+1}) \\
&\quad + \sum_{k=1}^{N} \left[\gamma^k \langle \delta^k, \nabla_{\mathbf{W}} \tilde{f}(\mathbf{W}^k) \rangle + \gamma^k \|\delta^k\|^2 \right]
\end{aligned}
\tag{5.115}
$$

We now take the expectation of the above inequality over the two sets of random variables – the stopping iteration R and the data instance η (similar to the setup from (5.66)-(5.70))

Note that $\delta^k = G(\eta^k; \mathbf{W}^k) - \nabla_{\mathbf{W}} f(\mathbf{W}^k)$ and recall the steps (5.66)-(5.68) (from the proof of Theorem 5.1) which were used to bound the expectations of the last two terms on the right hand side of (5.70). Replicating a similar set of steps here and using (5.107), we have the following

$$
\mathbb{E}_\eta \left(\sum_{k=1}^{N} \gamma^k \langle \delta^k, \nabla_{\mathbf{W}} \tilde{f}(\mathbf{W}^k) \rangle \right) = \sum_{k=1}^{N} \gamma^k \mathbb{E}_\eta \left(\langle \delta^k, \nabla_{\mathbf{W}} \tilde{f}(\mathbf{W}^k) \rangle | \eta^1, \ldots, \eta^{k-1} \right) = 0
\tag{5.116}
$$

$$\mathbb{E}_\eta\|\delta^k\|^2 = \frac{1}{B^2}\mathbb{E}_\eta\|\sum_{b=1}^{B}\delta^{b,k}\|^2 = \sum_{b=1}^{B}\mathbb{E}_\eta\|\delta^{b,k}\|^2 \leqslant \frac{e^{da}}{B} \tag{5.117}$$

$$\text{where} \quad e^{da} := \frac{d_x d_h}{16}\left[1 + \frac{\zeta d_x w_m}{4}\mu_x + \left(\frac{5\zeta}{16} - \frac{\zeta^2}{4}\right)(\zeta d_x w_m)^2\tau_x\right]$$

Using (5.116) and (5.117) after taking the expectation (over η) of (5.115) and using the fact that $f(\mathbf{W}^{N+1}) \leqslant f^*$, we have

$$\sum_{k=1}^{N}\left(\gamma^k - \frac{\mathcal{U}_{da}}{2}(\gamma^k)^2\right)\mathbb{E}_\eta\|\tilde{G}(\eta^k;\mathbf{W}^k)\|^2 \leqslant D_f + \frac{e^{da}}{B}\sum_{k=1}^{N}\gamma^k \tag{5.118}$$

where $D_f = f(\mathbf{W}^1) - f^*$.

Observe that the inequality to makes sense we need $\gamma^k < \frac{2}{\mathcal{U}_{da}}$. With this, and following the setup described using (5.71) for arriving at (5.73) using constant stepsizes $\gamma^k = \gamma < \frac{2}{\mathcal{U}_{da}}$ (refer to the proof of Theorem 5.1), we have the following

$$\left(\gamma - \frac{\mathcal{U}_{da}}{2}\gamma^2\right)\sum_{k=1}^{N}\mathbb{E}_\eta\|\tilde{G}(\eta^k;\mathbf{W}^k)\|^2 \leqslant D_f + \frac{e^{da}N\gamma}{B} \quad \text{and hence}$$

$$\mathbb{E}_{R,\eta}\|\tilde{G}(\eta^k;\mathbf{W}^k)\|^2 := \frac{1}{N}\sum_{k=1}^{N}\mathbb{E}_\eta\|\tilde{G}(\eta^k;\mathbf{W}^k)\|^2 \leqslant \frac{D_f + \frac{e^{da}N\gamma}{B}}{N(\gamma - \frac{\mathcal{U}_{da}}{2}\gamma^2)}$$
$$= \left(\frac{D_f}{N\gamma} + \frac{e^{da}}{B}\right)\frac{1}{1 - \frac{\mathcal{U}_{da}}{2}\gamma} \tag{5.119}$$

Recall that in all the derivations above η essentially corresponds to sampling a data instance $\mathbf{x} \in \mathcal{X}$ and then corrupting it to get $\tilde{\mathbf{x}}$. The above inequality governs the decay of $\tilde{G}(\eta^k;\mathbf{W}^k)$, however, we are interested in bounding the expectation of $\nabla_{\mathbf{W}}\tilde{f}(\eta;\mathbf{W})$. To that end, we have the following (the subscript of expectation is dropped to reduce the clutter),

$$\begin{aligned}\mathbb{E}_{R,\eta}\|\nabla_{\mathbf{W}}\tilde{f}(\mathbf{W})\|^2 &= \mathbb{E}_{R,\eta}\|\nabla_{\mathbf{W}}\tilde{f}(\mathbf{W}) + \tilde{G}(\eta^k;\mathbf{W}^k) - \tilde{G}(\eta^k;\mathbf{W}^k)\|^2 \\ &\leqslant \mathbb{E}_{R,\eta}\|\tilde{G}(\eta^k;\mathbf{W}^k)\|^2 + \mathbb{E}_{R,\eta}\|\nabla_{\mathbf{W}}\tilde{f}(\mathbf{W}) - \tilde{G}(\eta^k;\mathbf{W}^k)\|^2 \\ &\leqslant \left(\frac{D_f}{N\gamma} + \frac{e^{da}}{B}\right)\frac{1}{1 - \frac{\mathcal{U}_{da}}{2}\gamma} + \mathbb{E}_{R,\eta}\|\nabla_{\mathbf{W}}f(\mathbf{W}) - G(\eta^k;\mathbf{W}^k)\|^2\end{aligned} \tag{5.120}$$

where the last inequality uses (5.119) for the first term, and (5.109)(b) with $h_1 = \nabla_{\mathbf{W}}f(\mathbf{W})$ and $h_2 = G(\eta^k;\mathbf{W}^k)$ for the second term.

Recalling that $\mathbb{E}_\eta\|\nabla_{\mathbf{W}}f(\mathbf{W}) - G(\eta^k;\mathbf{W}^k)\|^2$ is simply $\mathbb{E}_\eta\|\delta^k\|^2$, and using (5.117) and the definition of e_γ^{da}, we finally get

$$\mathbb{E}_{R,\eta}\|\nabla_{\mathbf{W}}\tilde{f}(\mathbf{W})\|^2 \leqslant \frac{D_f}{N\gamma e_\gamma^{d_a}} + \frac{e^{d_a}}{B}\left(1+\frac{1}{e_\gamma^{d_a}}\right) \tag{5.121}$$

The bound in (5.121) is very large whenever the stepsizes lie closer to the left and right extremes of the allowed range $0 < \gamma < \frac{2}{\mathcal{U}_{d_a}}$. Hence, the optimal γ_o is given by the derivative of the right hand side above. Noting that $B \ll N$ for the regimes of interest, γ_o is given by

$$\begin{aligned}
&\mathcal{U}_{d_a} e^{d_a} N\gamma_o^2 + 2BD_f\mathcal{U}_{d_a}\gamma_o - 2BD_f = 0 \\
&\implies \gamma_o = -\frac{BD_f}{e^{d_a}N} + \sqrt{\left(\frac{BD_f}{e^{d_a}N}\right)^2 + \frac{2BD_f}{\mathcal{U}_{d_a}e^{d_a}N}} \approx \sqrt{\frac{2BD_f}{\mathcal{U}_{d_a}e^{d_a}N}}
\end{aligned} \tag{5.122}$$

where $D_f \approx f(\mathbf{W}^1)$. Observe that for reasonable values of w_m and d_h, $\mathcal{U}_{d_a} < 2$, and in tandem with $B \ll N$ this result is of practical use, instead of just being for theoretical interest. Clearly we need $\frac{2}{\mathcal{U}_{d_a}} > \frac{BD_f}{e^{d_a}N}$ for the above stepsize to make sense, and in the $B \ll N$ setting with reasonable w_m and d_h, this would be satisfied.

Proof of Corollary 5.9

Recall the proofs of Theorem 5.1 and Corollary 5.4 from Sections 5.9 and 5.9 respectively, and modifications required in Section 5.9 over the calculations from Section 5.9. The proof of this result entails a similar set of changes over the recipe and inequalities from Section 5.9. Hence, we just list down the changed set of inequalities and the resulting change in the final bound.
Using $p_R^k \leqslant p_R^{k+1}\forall k$, (5.119) would change as follows,

$$\begin{aligned}
\mathbb{E}_{R,\eta}\|\tilde{G}(\eta^k;\mathbf{W}^k)\|^2 &:= \sum_{k=1}^{N} p_R^k\left(\mathbb{E}_\eta\|\tilde{G}(\eta^k;\mathbf{W}^k)\|^2\right) \leqslant p_R^N\sum_{k=1}^{N}\left(\mathbb{E}_\eta\|\tilde{G}(\eta^k;\mathbf{W}^k)\|^2\right) \\
&\leqslant p_R^N\frac{D_f + \frac{e^{d_a}N\gamma}{B}}{\gamma - \frac{\mathcal{U}_{d_a}}{2}\gamma^2} = \left(\frac{D_f}{\gamma} + \frac{e^{d_a}N}{B}\right)\frac{p_R^N}{1-\frac{\mathcal{U}_{d_a}}{2}\gamma}
\end{aligned} \tag{5.123}$$

The rest of the derivation would follow the structure in (5.120)–(5.122), and hence we have the following

$$\mathbb{E}_{R,\eta}\|\nabla_{\mathbf{W}}\tilde{f}(\mathbf{W})\|^2 \leqslant \frac{p_R^N D_f}{\gamma e_\gamma^{d_a}} + \frac{e^{d_a}}{B}\left(1+\frac{p_R^N N}{e_\gamma^{d_a}}\right) \tag{5.124}$$

Re-computing the optimal stepsize from the above bound, following (5.122), we would still get (with $B \ll N$),

$$\gamma_o = -\frac{BD_f}{e^{d_a}N} + \sqrt{\left(\frac{BD_f}{e^{d_a}N}\right)^2 + \frac{2BD_f}{\mathcal{U}_{d_a}e^{d_a}N}} \approx \sqrt{\frac{2BD_f}{\mathcal{U}_{d_a}e^{d_a}N}} \tag{5.125}$$

Proof of Theorem 5.10

The expected gradients bound for DA RSG is (5.121). Now recall the large deviation estimate proof procedure for single layer RSG from Section 5.9, specifically the derivation of (5.90) and (5.91). Applying similar steps for (5.121), we get the following

$$\frac{D_f}{N\gamma e_\gamma^{d_a}} + \frac{e^{d_a}}{B}\left(1 + \frac{1}{e_\gamma^{d_a}}\right) \leqslant \epsilon\delta^{1/T} \tag{5.126}$$

Under the assumption that $SC \approx BN$, the two terms in the left hand side above decrease and increase respectively as N increases. Hence, we can balance them out by forcing each to be smaller than $\frac{\epsilon\delta^{1/T}}{2}$, which then gives

$$B \leqslant \frac{2e^{d_a}}{\epsilon\delta^{1/T}}\left(1 + \frac{1}{e_\gamma^{d_a}}\right) \qquad \text{and} \qquad N \geqslant \frac{2f(\mathbf{W}^1)}{\gamma e_\gamma^{d_a}\epsilon\delta^{1/T}} \tag{5.127}$$

Observe that $f(\mathbf{W}^1)$ is at most d_x which follows from the loss function of DA in (5.5) and the fact that $\mathbf{x} \in [0,1]^{d_x}$. Using this in the second inequality above and also the fact that $SC \approx BN$, we get

$$N \geqslant \max\left\{\frac{SC}{B}, \frac{2f(\mathbf{W}^1)}{\gamma e_\gamma^{d_a}\epsilon\delta^{1/T}}\right\} \tag{5.128}$$

Proof of Theorem 5.11

Observe that the setup of Algorithm 4 is to first pretrain the $L-1$ hidden layers, followed by running backpropagation through these layers using the outputs $\mathbf{y}$ (which correspond to the L^th layer). Hence, the analysis of expected gradients for this L-layered network (denoted by $\mathbf{W}^1, \ldots, \mathbf{W}^L$), starts from Theorem 5.8 where we bound the expected gradients of a single pretrained layer. Note the abuse of notation here: $\mathbf{W}^1, \ldots, \mathbf{W}^L$ denote the parameters of the network while $\mathbf{W}^{k,1}, \ldots, \mathbf{W}^{k,L}$ correspond to their estimates at k^{th} iteration of back propagation fine-tuning.

Consider the term $\|\tilde{g}(\eta; \mathbf{W})\| = \|P_{\mathbf{W}}(\mathbf{W}, \nabla_{\mathbf{W}}\mathcal{L}(\eta; \mathbf{W}), \gamma\|$, and the event

$$Pr(sup_\eta\|\tilde{g}(\eta; \mathbf{W})\| \leqslant \alpha) = 1 - Pr(sup_\eta\|\tilde{g}(\eta; \mathbf{W})\| \geqslant \alpha)$$

We intend to make this probability as small as possible. Using Jensen and Markov inequalities, we have the following

$$\sup_\eta \|\tilde{g}(\eta;\mathbf{W})\| \geqslant \mathbb{E}_\eta \|\tilde{g}(\eta;\mathbf{W})\| \geqslant \|\mathbb{E}_\eta \tilde{g}(\eta;\mathbf{W})\| \tag{5.129}$$

$$\begin{aligned}\Pr\left(\sup_\eta \|\tilde{g}(\eta;\mathbf{W})\|^2 \geqslant \|\mathbb{E}_\eta \tilde{g}(\eta;\mathbf{W})\|^2 \geqslant \alpha^2\right) &\leqslant \frac{\mathbb{E}_R \|\mathbb{E}\tilde{g}(\eta;\mathbf{W})\|^2}{\alpha^2} \\ &= \frac{1}{\alpha^2}\mathbb{E}_R \|\nabla_{\mathbf{W}} \tilde{f}(\eta;\mathbf{W})\|^2\end{aligned} \tag{5.130}$$

and using (5.18), we then get

$$\Pr(\sup_\eta \|\tilde{g}(\eta;\mathbf{W})\| \leqslant \alpha) \geqslant 1 - \frac{c_u}{\alpha^2} \qquad c_u = \frac{1}{C_\gamma}\left(\frac{D_f}{N\gamma} + \frac{\mathcal{V}_{da}}{B}\right) + \frac{\mathcal{V}_{da}}{B} \tag{5.131}$$

Consider a L-layered multi-layer network with lengths $d_0, d_1, \ldots, d_L$. Recall that d_0 and d_L correspond to the input ($\mathbf{x}$) and output ($\mathbf{y}$) layers respectively. Algorithm 4 pretrains the $L-1$ hidden layers (corresponding to $\mathbf{h}^1, \ldots, \mathbf{h}^{L-1}$). If the expected gradients bounds resulting from pretraining these layers are denoted by $c_u^1, \ldots, c_u^{L-1}$, then we have the following with $\delta_\alpha = \frac{1}{\alpha^2} max\{c_u^1, \ldots, c_u^{L-1}\}$

$$\Pr(\sup_\eta \|\tilde{g}_l(\eta;\mathbf{W})\| \leqslant \alpha) \geqslant 1 - \delta_\alpha \quad \text{for} \quad l = 1, \ldots, L-1 \tag{5.132}$$

We then operate in this setting where hidden layers are pretrained (refer Algorithm 4); α, δ_α and the corresponding pretraining hyper-parameters (refer to Theorem 5.8 and Corollary 5.10) are chosen such that the probability bound in (5.132) is satisfied.

We now setup the proof for bounding expected gradients for 'tuning' the network. Recall the *projected* back propagation setting used in Algorithm 4, where the gradients of $l-1^{th}$ take the following from based on the updates and gradients from l^{th} layer.

$$g^{ij}(\eta;\mathbf{W}^l) = h_q^{l-1} h_p^l (1 - h_p^l) \sum_{m=1}^{d_{l+1}} \tilde{g}_{l+1}^{m,i}(\eta;\mathbf{W}^{l+1}) W_{ij}^{l+1} \tag{5.133}$$

where $\mathbf{h}^l = \sigma(\mathbf{W}^l \mathbf{h}^{l-1})$ is the l^{th} hidden layer output, $i = 1, \ldots, d_l$ and $j = 1, \ldots, d_{l-1}$ ($\mathbf{h}^0 = \mathbf{x}$). Using this, and following the setup of proof from Theorem 5.1 we first compute the variance of the noisy gradients $g^{ij}(\eta;\mathbf{W}^l)$. Since the network is pretrained, using (5.132)

$$|g^{ij}(\eta;\mathbf{W}^l)| = |h_j^{l-1} h_i^l (1-h_i^l)\| \sum_{m=1}^{d_{l+1}} \tilde{g}_{l+1}^{m,i}(\eta;\mathbf{W}^{l+1})\| W_{ij}^{l+1}| \leqslant \frac{\alpha d_{l+1} w_m^l}{4} \tag{5.134}$$

and denoting $\delta^l = G_l(\eta; \mathbf{W}^l) - \nabla_{\mathbf{W}^l} f(\eta; \mathbf{W}^l)$, we have

$$\mathbb{E}_\eta \|\delta^l\|^2 \leqslant \frac{\alpha d_{l-1} d_l d_{l+1} w_m^l}{4} \tag{5.135}$$

Following the recipe from Theorem 5.1, we then compute the Lipschitz constants for the hidden layer gradients $\tilde{g}_l(\eta; \mathbf{W}^l)$. Using (5.133) and rearranging few terms, we have

$$\begin{aligned}
&\|g_l^{ij}(\eta; \mathbf{W}) - g_l^{ij}(\eta; \hat{\mathbf{W}})\| \\
&\qquad \leqslant |h_j^{l-1} \sigma_i(\mathbf{W}^l \mathbf{h}^{l-1})(1 - \sigma_i(\mathbf{W}^l \mathbf{h}^{l-1})) \sum_{m=1}^{d_{l+1}} \tilde{g}_{l+1}^{m,i}(\eta; \mathbf{W}^{l+1}) W_{ij}^{l+1} \\
&\qquad - h_j^{l-1} \sigma_i(\hat{\mathbf{W}}^l \mathbf{h}^{l-1})(1 - \sigma_i(\hat{\mathbf{W}}^l \mathbf{h}^{l-1})) \sum_{m=1}^{d_{l+1}} \tilde{g}_{l+1}^{m,i}(\eta; \mathbf{W}^{l+1}) W_{ij}^{l+1}| \\
&\qquad \leqslant |\sigma_i(\mathbf{W}^l \mathbf{h}^{l-1})(1 - \sigma_i(\mathbf{W}^l \mathbf{h}^{l-1})) - \sigma_i(\hat{\mathbf{W}}^l \mathbf{h}^{l-1})(1 - \sigma_i(\hat{\mathbf{W}}^l \mathbf{h}^{l-1}))| \\
&\qquad \sum_{m=1}^{d_{l+1}} |\tilde{g}_{l+1}^{m,i}(\eta; \mathbf{W}^{l+1})| |W_{ij}^{l+1}| \leqslant \frac{\alpha d_{l+1} w_m^l}{10} |W_{ij}^l - \hat{W}_{ij}^l|
\end{aligned} \tag{5.136}$$

Using (5.135) and (5.136), we now follow the recipe for the proofs of Theorem 5.1 and Theorem 5.8 to derive the expected gradients. Recall the update steps from Algorithm 4, which essentially is back propagating the errors from the output layer L to the first/input layer. Within a single iteration of the backprop all the layers are updated i.e., $\{\mathbf{W}^{k+1,l}\} \leftarrow \{\mathbf{W}^{k,l}\}\ \forall l$. Further, the objective for each of the L updates within this iteration is $f(\mathbf{W}^l) = \mathbb{E}\|\mathbf{h}^l - \sigma(\mathbf{W}^l \mathbf{h}^{l-1})\|^2$ ($\mathbf{h}^L = \mathbf{y}$, $\mathbf{h}^0 = \mathbf{x}$).

Using this intuition and recalling that is no pretraining for the output (L^{th}) layer, we start with the following inequality concerning the final layer (based on (5.61)),

$$f(\mathbf{W}^{k+1,l}) \leqslant f(\mathbf{W}^{k,l}) + \langle \nabla_{\mathbf{W}^l} f(\mathbf{W}^{k,l}), \mathbf{W}^{k+1,l} - \mathbf{W}^{k,l} \rangle + \frac{13}{16} \|\mathbf{W}^{k+1,l} - \mathbf{W}^{k,l}\|^2 \tag{5.137}$$

Applying the same set of steps as in (5.61)–(5.63), and recalling that $\delta^{k,L} = G(\eta; \mathbf{W}^L) - \nabla_{\mathbf{W}^L} f(\mathbf{W}^L)$ we get

$$f(\mathbf{W}^{k+1,L}) \leqslant f(\mathbf{W}^{k,L}) - \left(\gamma_L^k - \frac{13}{16}(\gamma_L^k)^2\right) \|\nabla_{\mathbf{W}^L} f(\mathbf{W}^{k,L})\|^2 - \left(\gamma_L^k - \frac{13}{8}(\gamma_L^k)^2\right)\langle \nabla_{\mathbf{W}^L} f(\mathbf{W}^{k,L}), \delta^{k,L}\rangle + \frac{13}{16}(\gamma_L^k)^2 \|\delta^{k,L}\|^2 \tag{5.138}$$

Once the L^{th} layer is updated, the remaining $L-1$ layers are updated from top to bottom (refer Algorithm 4).

Based on the set of inequalities derived in (5.111)–(5.114), the corresponding bounds on the layer-wise objectives $f(\mathbf{W}^l)$ will be as follows

$$f(\mathbf{W}^{k+1,l}) \leqslant f(\mathbf{W}^{k,l}) - \left(\gamma_l^k - \frac{\alpha d_{l+1} w_m^l}{20}(\gamma_l^k)^2\right) \|\tilde{G}(\eta^k; \mathbf{W}^{k,l})\|^2 + \gamma_l^k \langle \delta^{k,l}, \nabla_{\mathbf{W}^l} \tilde{f}(\mathbf{W}^{k,l})\rangle + \gamma_l^k \|\delta^{k,l}\|^2 \quad ; \quad l = L-1, \ldots, 1 \tag{5.139}$$

where the Lipschitz bound from (5.136) was used and $\delta^{k,l} = G(\eta; \mathbf{W}^l) - \nabla_{\mathbf{W}^l} f(\mathbf{W}^l)$.

Observe that the final layer has no constraints imposed on the parameters W_{ij}^L, and so $\nabla_{\mathbf{W}^L} \tilde{f}(\mathbf{W}^{k,L}) = \nabla_{\mathbf{W}^L} f(\mathbf{W}^{k,L})$ and $\tilde{G}(\eta^k; \mathbf{W}^{k,l}) = G(\eta^k; \mathbf{W}^{k,L})$. Denoting $F(\mathbf{W}^k) := \sum_{l=1}^L f(\mathbf{W}^{k,l})$, and adding up (5.138) and (5.139), we get

$$\begin{aligned} F(\mathbf{W}^{k+1}) \leqslant\ & F(\mathbf{W}^k) - \left(\gamma_L^k - \frac{13}{16}(\gamma_L^k)^2\right) \|\nabla_{\mathbf{W}^L} f(\mathbf{W}^{k,L})\|^2 \\ & - \sum_{l=1}^{L-1} \left(\gamma_l^k - \frac{\alpha d_{l+1} w_m^l}{20}(\gamma_l^k)^2\right) \|\tilde{G}(\eta^k; \mathbf{W}^{k,l})\|^2 \\ & - \left(\gamma_L^k - \frac{13}{8}(\gamma_L^k)^2\right)\langle \nabla_{\mathbf{W}^L} f(\mathbf{W}^{k,L}), \delta^{k,L}\rangle \\ & + \sum_{l=1}^{L-1} \gamma_l^k \langle \delta^{k,l}, \nabla_{\mathbf{W}^l} \tilde{f}(\mathbf{W}^{k,l})\rangle + \frac{13}{16}(\gamma_L^k)^2 \|\delta^{k,L}\|^2 + \sum_{l=1}^{L-1} \gamma_l^k \|\delta^{k,l}\|^2 \end{aligned} \tag{5.140}$$

We restrict to the constant stesizes case, i.e., $\gamma_l^k = \gamma_l \forall k$ (for a given layer). Now, summing the above inequality over the N iterations and rearranging terms, we get

$$\begin{aligned}
&\left(\gamma_L - \frac{13}{16}(\gamma_L)^2\right)\sum_{k=1}^{N}\|\nabla_{\mathbf{W}^L} f(\mathbf{W}^{k,L})\|^2 \\
&+\sum_{l=1}^{L-1}\left(\gamma_l - \frac{\alpha d_{l+1} w_m^l}{20}(\gamma_l)^2\right)\sum_{k=1}^{N}\|\tilde{G}(\eta^k;\mathbf{W}^{k,l})\|^2 \\
&\leqslant F(\mathbf{W}^1) - F^* - \left(\gamma_L - \frac{13}{8}(\gamma_L)^2\right)\sum_{k=1}^{N}\langle\nabla_{\mathbf{W}^L} f(\mathbf{W}^{k,L}),\delta^{k,L}\rangle \\
&+\sum_{l=1}^{L-1}\gamma_l\sum_{k=1}^{N}\langle\delta^{k,l},\nabla_{\mathbf{W}^l}\tilde{f}(\mathbf{W}^{k,l})\rangle + \frac{13}{16}(\gamma_L)^2\sum_{k=1}^{N}\|\delta^{k,L}\|^2 + \sum_{l=1}^{L-1}\gamma_l\sum_{k=1}^{N}\|\delta^{k,l}\|^2
\end{aligned} \tag{5.141}$$

where $\mathbf{W}^1$ represents the initial estimate – $\mathbf{W}^{1,L}$ for the final layer, and the pretrained estimates for the remaining $L-1$ layers.

We now take the expectation of the above inequality with η, followed by the stopping iteration R. Recall the set of inequalities from (5.66) and (5.116). Replicating a similar set of analysis, the terms involving $\langle\cdot,\cdot\rangle$ from the right hand side above will vanish. Further, from (5.67)–(5.69) and (5.117), the last two terms in the above inequality will reduce to

$$\begin{aligned}
\mathbb{E}_\eta\left(\frac{13}{16}(\gamma_L)^2\sum_{k=1}^{N}\|\delta^{k,L}\|^2\right) &\leqslant \frac{13 d_{L-1} d_L}{256B} N\gamma_L^2 \\
\mathbb{E}_\eta\left(\sum_{l=1}^{L-1}\gamma_l\sum_{k=1}^{N}\|\delta^{k,l}\|^2\right) &\leqslant \frac{N\alpha}{4B}\sum_{l=1}^{L-1}\gamma_l d_{l-1} d_l d_{l+1} w_m^l
\end{aligned} \tag{5.142}$$

The second inequality uses (5.135). Here the size of mini-batch B is the same across all layers. Using (5.142) after applying the expectation to (5.137), we have the following

$$\begin{aligned}
&\left(\gamma_L - \frac{13}{16}(\gamma_L)^2\right)\sum_{k=1}^{N}\mathbb{E}\|\nabla_{\mathbf{W}^L} f(\mathbf{W}^{k,L})\|^2 \\
&+\sum_{l=1}^{L-1}\left(\gamma_l - \frac{\alpha d_{l+1} w_m^l}{20}(\gamma_l)^2\right)\sum_{k=1}^{N}\mathbb{E}\|\tilde{G}(\eta^k;\mathbf{W}^{k,l})\|^2 \\
&\leqslant D_F + \frac{N}{4B}\left(\frac{13 d_{L-1} d_L \gamma_L^2}{64} + \alpha\sum_{l=1}^{L-1}\gamma_l d_{l-1} d_l d_{l+1} w_m^l\right)
\end{aligned} \tag{5.143}$$

To allow for 'simpler' summarization of the above inequality before taking the expectation with respect to the stopping iteration, denote

$$\tilde{\gamma} = \min\left[\gamma_L - \frac{13}{16}(\gamma_L)^2, \gamma_l - \frac{\alpha d_{l+1} w_m^l}{20}(\gamma_l)^2\right] \quad l = L-1, \ldots, 1 \tag{5.144}$$

With this, we have

$$\begin{aligned}
&\tilde{\gamma}\sum_{k=1}^{N}\left(\mathbb{E}\|\nabla_{\mathbf{W}^L} f(\mathbf{W}^{k,L})\|^2 + \sum_{l=1}^{L-1}\mathbb{E}\|\tilde{G}(\eta^k; \mathbf{W}^{k,l})\|^2\right) \\
&\qquad \leqslant \left(\gamma_L - \frac{13}{16}(\gamma_L)^2\right)\sum_{k=1}^{N}\mathbb{E}\|\nabla_{\mathbf{W}^L} f(\mathbf{W}^{k,L})\|^2 \\
&\qquad\qquad + \sum_{l=1}^{L-1}\left(\gamma_l - \frac{\alpha d_{l+1} w_m^l}{20}(\gamma_l)^2\right)\sum_{k=1}^{N}\mathbb{E}\|\tilde{G}(\eta^k; \mathbf{W}^{k,l})\|^2
\end{aligned} \tag{5.145}$$

Using the decomposition of $\mathbb{E}_R\|\tilde{f}(\mathbf{W}^{k,l})\|^2$ based on (5.120), the above inequality gives

$$\begin{aligned}
\mathbb{E}_{R,\eta}\|\nabla_{\mathbf{W}^1,\ldots,\mathbf{W}^L}\tilde{f}(\mathbf{W})\|^2 &:= \sum_{l=1}^{L}\mathbb{E}_{R,\eta}\|\nabla_{\mathbf{W}^l}\tilde{f}(\mathbf{W})\|^2 \\
&= \mathbb{E}_{R,\eta}\|\nabla_{\mathbf{W}^L} f(\mathbf{W})\|^2 + \sum_{l=1}^{L-1}\mathbb{E}_{R,\eta}\|\nabla_{\mathbf{W}^l}\tilde{f}(\mathbf{W})\|^2 \\
&= \mathbb{E}_{R,\eta}\|\nabla_{\mathbf{W}^L} f(\mathbf{W})\|^2 + \sum_{l=1}^{L-1}\mathbb{E}_{R,\eta}\|\tilde{G}(\eta^k; \mathbf{W}^{k,l})\|^2 \\
&\qquad + \sum_{l=1}^{L-1}\|\nabla_{\mathbf{W}^l}\tilde{f}(\mathbf{W}) - \tilde{G}(\eta^k; \mathbf{W}^{k,l})\|^2 \\
&= \frac{1}{N}\sum_{k=1}^{N}\mathbb{E}_{\eta}\|\nabla_{\mathbf{W}^L} f(\mathbf{W}^{k,L})\|^2 + \frac{1}{N}\sum_{k=1}^{N}\sum_{l=1}^{L-1}\mathbb{E}_{\eta}\|\tilde{G}(\eta^k; \mathbf{W}^{k,l})\|^2 \\
&\qquad + \sum_{l=1}^{L-1}\|\nabla_{\mathbf{W}^l}\tilde{f}(\mathbf{W}) - \tilde{G}(\eta^k; \mathbf{W}^{k,l})\|^2
\end{aligned} \tag{5.146}$$

where the last inequality uses the fact that the stopping iteration is uniformly chosen over the $k = 1, \ldots, N$ iterations and (5.135).

Clearly, by the definition of $F(\mathbf{W})$ from above, we see that $D_F = D_f$ (since $f^l(\mathbf{W}^{1,l}) = 0$ for $l = 1, \ldots, L-1$). Using this, (5.137) and (5.145), we finally have

$$
\begin{aligned}
&\mathbb{E}_{R,\eta}\|\nabla_{\mathbf{W}^1,\ldots,\mathbf{W}^L}\tilde{f}(\mathbf{W})\|^2 \\
&\quad\leqslant \frac{D_f}{Ne_\gamma^m}+\frac{1}{4Be_\gamma^m}\left(\frac{13d_{L-1}d_L\gamma_L^2}{64}+\alpha\sum_{l=1}^{L-1}\gamma_l d_{l-1}d_l d_{l+1}w_m^l\right) \\
&\quad= \frac{1}{e_\gamma^m}\left(\frac{D_f}{N}+\frac{1}{B}(e_L^m+\alpha\sum_{l=1}^{L-1}e_l^m)\right) \\
&\text{where}\quad e_l^m=\frac{\gamma^l}{4}d_{l-1}d_l d_{l+1}w_m^l\quad(l=1,\ldots,L-1)\quad\text{and}\quad e_L^m=\frac{13d_{L-1}d_L\gamma_L^2}{256}
\end{aligned}
\tag{5.147}
$$

Proof of Corollary 5.13

The proof of this result directly follows from (5.23), which is presented below for reference.

$$
\begin{aligned}
&\mathbb{E}\|\nabla_{\mathbf{W}}\tilde{f}(\mathbf{W}^R)\|^2\leqslant\frac{1}{e_\gamma^m}\left(\frac{D_f}{N}+\frac{1}{B}(e_L^m+\sum_{l=1}^{L-1}\alpha_l e_l^m)\right) \\
&\text{with}\qquad e_\gamma^m=\min\left\{\gamma_L-\frac{13}{16}(\gamma_L)^2,\gamma_l-\frac{\alpha_l d_{l+1}w_m^l}{20}(\gamma_l)^2\right\}
\end{aligned}
\tag{5.148}
$$

Under the assumption that $\gamma_l=\gamma\forall l$, and whenever $\alpha_l d_{l+1}<\frac{65}{4}$, we have

$$
\begin{aligned}
&\frac{\alpha_l d_{l+1}}{20}\gamma^2<\frac{13}{16}\gamma^2\quad\text{for any }\gamma \\
&\quad\text{and hence}\quad e_\gamma^m=\gamma-\frac{13}{16}\gamma^2\quad\text{will be a constant}
\end{aligned}
\tag{5.149}
$$

Now, for a given B and N, and approximating $D_f\approx d_L$, the term that will be different across the networks $\mathcal{D}:=(\alpha_l,d_l)$ in the decay bound from (5.148) is the following (using the definition of e_l^ms from Theorem 5.11),

$$
\frac{13\gamma^2}{256}d_{L-1}d_L+\frac{\gamma}{4}\sum_{l=1}^{L-1}\alpha_l d_{l-1}d_l d_{l+1}
\tag{5.150}
$$

Now all the networks that satisfy the following condition, will have the same (5.150), and hence the decay bound would be the same i.e., they all are equivalent according to Definition 5.12.

$$
\alpha_l d_{l-1}d_l d_{l+1}=\frac{1}{\Psi_l}\quad\text{for a given }\Psi_1,\ldots,\Psi_{L-1},\ d_L\text{ and }d_{L-1}
\tag{5.151}
$$

Proof of Corollary 5.14

The proof follows by comparing the decay bound from (5.22) for the two networks $\mathcal{D}_s$ (depth L_s) and $\mathcal{D}_t$ (depth L_t). Recall that d_0 and d_L are given. $d_t = \frac{d_s}{1+\delta_{st}}$ with $L_t > L_s$ and the pretraining goodness is α for both (for all the layers) networks. Since we are interested in the regime where the taller network $\mathcal{D}_t$ is better than the shorter one $\mathcal{D}_s$, we have the following using (5.22),

$$\begin{aligned}\frac{1}{e_\gamma^m}\left(\frac{d_L}{N}+\frac{1}{B}\left(\frac{13d_sd_L\gamma^2}{256}+\frac{\alpha\gamma w}{4}(d_s^2d_L+d_s^2d_0)+\frac{\alpha\gamma w}{4}(L_s-3)d_s^3\right)\right)\\ \geqslant \frac{1}{e_\gamma^m}\left(\frac{d_L}{N}+\frac{1}{B}\left(\frac{13d_td_L\gamma^2}{256}+\frac{\alpha\gamma w}{4}(d_t^2d_L+d_t^2d_0)+\frac{\alpha\gamma w}{4}(L_t-3)d_t^3\right)\right)\end{aligned} \tag{5.152}$$

where $e_\gamma^m = \gamma - \frac{13}{16}\gamma^2$ and the definition of e_l^ms was used from Theorem 5.11.

Rearranging and cancelling several terms will give

$$\begin{aligned}\frac{13d_L\gamma^2}{256}(d_s-d_t)+\frac{\alpha\gamma w}{4}(d_L+d_0)(d_s^2-d_t^2)\\ \geqslant \frac{\alpha\gamma w}{4}\left((L_t-3)d_t^3-(L_s-3)d_s^3\right)\end{aligned} \tag{5.153}$$

Using $d_s = d_t(1+\delta_{st})$ in the above inequality, we get

$$\begin{aligned}\frac{13d_L\gamma}{64}\delta_{st}+\alpha w\delta_{st}(2+\delta_{st})(d_L+d_0)d_t\\ \geqslant \alpha w d_t^2\left((L_t-3)-(L_s-3)(1+\delta_{st})^3\right)\end{aligned} \tag{5.154}$$

which then gives

$$L_t-3 \leqslant (L_s-3)(1+\delta_{st})^3+\frac{1}{\alpha w d_t^2}\left(\frac{13d_L\gamma}{64}\delta_{st}+\alpha w\delta_{st}(2+\delta_{st})(d_L+d_0)d_t\right) \tag{5.155}$$

Recall that $\delta_{st} > 0$, and so any L_t that satisfies the following upper bound will satisfy the above inequality,

$$\begin{aligned}L_t &\leqslant (L_s-3)(1+3\delta_{st})+3+\delta_{st}\left(\frac{13d_L\gamma}{64\alpha w d_t^2}+\frac{(1+\delta_{st})(d_L+d_0)}{d_t}\right)\\ &= (L_s-3)(1+3\delta_{st})+3\\ &\quad+\delta_{st}\left(\frac{\frac{13d_td_L\gamma^2}{256}}{\alpha\frac{\gamma w}{4}d_t^3}+(1+\delta_{st})\left(\frac{\frac{\gamma w}{4}d_t^2d_L}{\frac{\gamma w}{4}d_t^3}+\frac{\frac{\gamma w}{4}d_t^2d_0}{\frac{\gamma w}{4}d_t^3}\right)\right)\end{aligned} \tag{5.156}$$

where in the second inequality several terms are adjusted to get the expression

into a form of interest (that will be clear shortly). Now using the fact that $d_t = \frac{d_s}{1+\delta_{st}}$, and the definitions of e_l^m from Theorem 5.11, we finally have (recall that under the assumptions $e_2^m = \ldots = e_{L-2}^m$ for $\mathcal{D}_s$),

$$\begin{aligned} L_t &\leqslant L_s + 3\delta_{st}(L_s - 3) + \delta_{st}(1+\delta_{st})^2 \left(\frac{\frac{13 d_s d_L \gamma^2}{256}}{\alpha \frac{\gamma w}{4} d_s^3} + \frac{\frac{\gamma w}{4} d_s^2 d_L}{\frac{\gamma w}{4} d_s^3} + \frac{\frac{\gamma w}{4} d_s^2 d_0}{\frac{\gamma w}{4} d_s^3} \right) \\ &= L_s + 3\delta_{st}(L_s - 3) + \delta_{st}(1+\delta_{st})^2 \left(\frac{e_L^m}{\alpha e_2^m} + \frac{e_{L-1}^m}{e_2^m} + \frac{e_1^m}{e_2^m} \right) \end{aligned} \tag{5.157}$$

Proof of Theorem 5.15

Inducing dropout into the setting of Corollary 5.11 corresponds to adjusting the terms in (5.143)–(5.147). The arguments follow through what was presented in the proof of Corollary 5.7 in Section 5.9.

Those same observations hold for the multi-layer setting as well, and we follow the derivative arguments presented for 1-NN above. Since the setting for this results, assumes that the network has been pretrained layer-wise, the results from Corollary 5.11 will hold. We now adapt them (specifically (5.137)–(5.147)) to this multi-layer dropout case.

Firstly the constants $e_1^m, \ldots, e_L^m$ would change as follows (refer to the statement of Corollary 5.11),

$$\begin{aligned} e_l^m; &\quad \frac{\gamma^l}{4} d_{l-1} d_l d_{l+1} w_m^l \to \frac{\gamma^l}{4} \zeta_{l-1} \zeta_l \zeta_{l+1} d_{l-1} d_l d_{l+1} w_m^l \quad (l = 1, \ldots, L-2) \\ e_{L-1}^m; &\quad \frac{\gamma^l}{4} d_{L-2} d_{L-1} d_L w_m^l \to \frac{\gamma^l}{4} \zeta_{L-2} \zeta_{L-1} d_{L-2} d_{L-1} d_L w_m^l \\ e_L^m; &\quad \frac{13 d_{L-1} d_L \gamma_L^2}{256} \to \frac{13 \zeta_{L-1} d_{L-1} d_L \gamma_L^2}{256} \end{aligned} \tag{5.158}$$

Secondly, the left hand side in (5.143) changes to

$$\begin{aligned} &\zeta_L \left(\gamma_L - \frac{13}{16} (\gamma_L)^2 \right) \sum_{k=1}^{N} \mathbb{E} \| \nabla_{\mathbf{W}^l} f(\mathbf{W}^{k,L}) \|^2 \\ &\quad + \sum_{l=1}^{L-1} \zeta_{l-1} \zeta_l \left(\gamma_l - \frac{\alpha d_{l+1} w_m^l}{20} (\gamma_l)^2 \right) \sum_{k=1}^{N} \mathbb{E} \| \tilde{G}(\eta^k; \mathbf{W}^{k,l}) \|^2 \end{aligned} \tag{5.159}$$

Note that this basically follows from the correction for effective number of times W_{ij}'s are getting updated in the dropout setting (refer to the discussion about

(5.93) and (5.94) from Section 5.9). Using the fact that $\zeta_l^2 < \zeta_l$ and denoting $\underline{\zeta} = \min_l \zeta_l$, we get

$$\begin{aligned}
&\underline{\zeta}^2 \left(\left(\gamma_L - \frac{13}{16}(\gamma_L)^2 \right) \sum_{k=1}^{N} \mathbb{E}\|\nabla_{\mathbf{W}^L} f(\mathbf{W}^{k,L})\|^2 \right) \\
&\quad + \underline{\zeta}^2 \left(\sum_{l=1}^{L-1} \left(\gamma_l - \frac{\alpha d_{l+1} w_m^l}{20}(\gamma_l)^2 \right) \sum_{k=1}^{N} \mathbb{E}\|\tilde{G}(\eta^k; \mathbf{W}^{k,l})\|^2 \right) < \\
&\zeta_L \left(\gamma_L - \frac{13}{16}(\gamma_L)^2 \right) \sum_{k=1}^{N} \mathbb{E}\|\nabla_{\mathbf{W}^L} f(\mathbf{W}^{k,L})\|^2 \\
&\quad + \sum_{l=1}^{L-1} \zeta_{l-1}\zeta_l \left(\gamma_l - \frac{\alpha d_{l+1} w_m^l}{20}(\gamma_l)^2 \right) \sum_{k=1}^{N} \mathbb{E}\|\tilde{G}(\eta^k; \mathbf{W}^{k,l})\|^2
\end{aligned} \tag{5.160}$$

Lastly, the hyper-parameter e_γ^m would also need to be changed, but assuming that the stepsizes γ^ls and α are reasonably small, so that, e_γ^m would be approximately the same as in the non-dropout case. If this is not the case, then the analysis and the inequalities derived become very messy, with a lot of clutter, and in the end, nothing new to infer from compared to the case where e_γ^m would simply depend on the γ^ls instead of d_ls. Hence, we use the same e_γ^m as was used in Theorem 5.11.

With the changes in (5.158) and (5.160), and using $\bar{\zeta} = \max_l \zeta_l$, we finally have (from (5.147))

$$\begin{aligned}
\mathbb{E}_{R,\eta}\|\nabla_{\mathbf{W}^1,\ldots,\mathbf{W}^L} \tilde{f}(\mathbf{W})\|^2 &\leqslant \frac{D_F}{N\underline{\zeta}^2 e_\gamma^m} + \frac{1}{4B\underline{\zeta}^2 e_\gamma^m} \left(\frac{13\zeta_{L-1} d_{L-1} d_L \gamma_L^2}{64} \right) \\
&\quad + \frac{1}{4B\underline{\zeta}^2 e_\gamma^m} \left(\alpha \gamma_l \zeta_{L-2}\zeta_{L-1} d_{L-2} d_{L-1} d_L w_m^{L-1} \right) \\
&\quad + \frac{1}{4B\underline{\zeta}^2 e_\gamma^m} \left(\alpha \sum_{l=1}^{L-2} \gamma_l \zeta_{l-1}\zeta_l\zeta_{l+1} d_{l-1} d_l d_{l+1} w_m^l \right) \\
&\leqslant \frac{D_F}{N\underline{\zeta}^2 e_\gamma^m} + \frac{1}{4B\underline{\zeta}^2 e_\gamma^m} \left(\frac{13\bar{\zeta} d_{L-1} d_L \gamma_L^2}{64} \right) \\
&\quad + \frac{1}{4B\underline{\zeta}^2 e_\gamma^m} \left(\alpha \gamma_l \bar{\zeta}^2 d_{L-2} d_{L-1} d_L w_m^{L-1} + \alpha \sum_{l=1}^{L-2} \gamma_l \bar{\zeta}^3 d_{l-1} d_l d_{l+1} w_m^l \right)
\end{aligned} \tag{5.161}$$

Using the definitions of $e_l^m \forall l$ (refer to the statement of Theorem 5.11) reduces to

$$\mathbb{E}_{R,\eta}\|\nabla_{\mathbf{W}}\tilde{f}(\mathbf{W})\|^2 \leqslant \frac{1}{e_\gamma^m}\left(\frac{D_f}{N\underline{\zeta}^2}+\frac{1}{B}\left(\frac{\bar{\zeta}e_L^m}{\underline{\zeta}^2}+\frac{\alpha\bar{\zeta}^2e_{L-1}^m}{\underline{\zeta}^2}+\frac{\alpha\bar{\zeta}^3}{\underline{\zeta}^2}\sum_{l=1}^{L-2}e_l^m\right)\right) \tag{5.162}$$

which further reduces to the following, using that fact that $\bar{\zeta}^3 < \bar{\zeta}^2$ and recalling the definition of $\zeta_{\mathcal{B}}$ from the Theorem statement,

$$\mathbb{E}_{R,\eta}\|\nabla_{\mathbf{W}}\tilde{f}(\mathbf{W})\|^2 \leqslant \frac{1}{e_\gamma^m}\left(\frac{D_f}{N\underline{\zeta}^2}+\frac{1}{B}\left(\frac{\zeta_{\mathcal{B}}e_L^m}{\underline{\zeta}}+\alpha\zeta_{\mathcal{B}}^2e_{L-1}^m+\alpha\zeta_{\mathcal{B}}^2\sum_{l=1}^{L-2}e_l^m\right)\right) \tag{5.163}$$

Proof of Corollary 5.16

The proof corresponds to some adjustments in (5.158)–(5.163) from Section 5.9. We list them here, followed by the changed bound.
The constants $e_1^m,\ldots,e_L^m$ would change as follows,

$$\begin{aligned}
e_l^m;&\quad \frac{\gamma^l}{4}d_{l-1}d_ld_{l+1}w_m^l \to \frac{\gamma^l}{4}\zeta^3d_{l-1}d_ld_{l+1}w_m^l \quad (l=1,\ldots,L-2)\\
e_{L-1}^m;&\quad \frac{\gamma^l}{4}d_{L-2}d_{L-1}d_Lw_m^l \to \frac{\gamma^l}{4}\zeta^2d_{L-2}d_{L-1}d_Lw_m^l\\
e_L^m;&\quad \frac{13d_{L-1}d_L\gamma_L^2}{256} \to \frac{13\zeta d_{L-1}d_L\gamma_L^2}{256}
\end{aligned} \tag{5.164}$$

The left hand side in (5.143) changes to

$$\begin{aligned}
&\zeta\left(\gamma_L-\frac{13}{16}(\gamma_L)^2\right)\sum_{k=1}^{N}\mathbb{E}\|\nabla_{\mathbf{W}^L}f(\mathbf{W}^{k,L})\|^2\\
&\quad+\zeta^2\sum_{l=1}^{L-1}\left(\gamma_l-\frac{\alpha d_{l+1}w_m^l}{20}(\gamma_l)^2\right)\sum_{k=1}^{N}\mathbb{E}\|\tilde{G}(\eta^k;\mathbf{W}^{k,l})\|^2
\end{aligned} \tag{5.165}$$

Using these and the fact that $\zeta^2 < \zeta$, we have (from (5.147))

$$\begin{aligned}
\mathbb{E}_{R,\eta}\|\nabla_{\mathbf{W}^1,\ldots,\mathbf{W}^L}\tilde{f}(\mathbf{W})\|^2 &\leqslant \frac{D_F}{N\zeta^2e_\gamma^m}+\frac{1}{4B\zeta^2e_\gamma^m}\left(\frac{13\zeta d_{L-1}d_L\gamma_L^2}{64}\right)\\
&\quad+\frac{1}{4B\zeta^2e_\gamma^m}\left(\alpha\gamma_l\zeta^2d_{L-2}d_{L-1}d_Lw_m^{L-1}+\alpha\zeta^3\sum_{l=1}^{L-2}\gamma_ld_{l-1}d_ld_{l+1}w_m^l\right)
\end{aligned} \tag{5.166}$$

which then reduces to

$$\mathbb{E}_{R,\eta}\|\nabla_{\mathbf{W}}\tilde{f}(\mathbf{W})\|^2 \leqslant \frac{1}{e_\gamma^m}\left(\frac{D_f}{N\zeta^2}+\frac{1}{B}\left(\frac{e_L^m}{\zeta}+\alpha e_{L-1}^m+\alpha\zeta\sum_{l=1}^{L-2}e_l^m\right)\right) \tag{5.167}$$

Proof of Theorem 5.17

The proof for this result is similar to that of Theorem 5.10 from Section 5.9. The only difference is instead of using the expected gradients for DA, we will now be using the bound for multilayer RSG from (5.167), and everything else follows the setup from Section 5.9.

Proof of Corollary 5.18

The proof follows by working with the terms in the decay bound from (5.31) in Theorem 5.16. Using the definitions of e_l^ms from Theorem 5.11, and the assumption that $\gamma^l = \gamma$, $w_m^l = w > 0 \forall l$ and $D_f \approx d_L$, (5.31) reduces to

$$\begin{aligned}\mathbb{E}\|\nabla_{\mathbf{W}}\tilde{f}(\mathbf{W}^R)\|^2 &\lesssim \frac{d_L}{e_\gamma^m N\zeta^2} \\ &+ \frac{1}{e_\gamma^m B}\left(\frac{13\gamma^2}{256\zeta}d_{L-1}d_L + \zeta\sum_{l=1}^{L-2}\frac{\alpha\gamma w}{4}d_{l-1}d_l d_{l+1} + \frac{\alpha\gamma w}{4}d_{L-2}d_{L-1}d_L\right)\end{aligned} \tag{5.168}$$

Using the fact that $\zeta \leqslant 1$, the bound in (5.168) can be further reduced to

$$\mathbb{E}\|\nabla_{\mathbf{W}}\tilde{f}(\mathbf{W}^R)\|^2 \lesssim \frac{1}{e_\gamma^m\zeta^2}\left(\frac{d_L}{N} + \frac{13d^2\gamma^2}{256B}\right) + \frac{\zeta\alpha\gamma w}{4e_\gamma^m B}\left(d_0d^2 + (L-3)d^3 + d^2d_L\right) \tag{5.169}$$

Balancing the terms involving ζ, we then have the following (using the fact that $\zeta \in (0,1]$)

$$\begin{aligned}&\frac{1}{e_\gamma^m\zeta^2}\left(\frac{d_L}{N} + \frac{13d^2\gamma^2}{256B}\right) \approx \frac{\zeta\alpha\gamma w}{4e_\gamma^m B}\left(d_0d^2 + (L-3)d^3 + d^2d_L\right) \\ &\text{and hence the optimal } \zeta := \text{median}\left\{0, \sqrt[3]{\frac{C_1}{C_2}}, 1\right\} \\ \text{where}\quad &C_1 = \left(\frac{d_L}{N} + \frac{13d^2\gamma^2}{256B}\right)\quad C_2 = \frac{\alpha\gamma w}{4B}\left(d_0d^2 + (L-3)d^3 + d^2d_L\right)\end{aligned} \tag{5.170}$$

Proof of Corollary 5.19

The proof for this result follows very closely to that of Corollary 5.13 from Section 5.9. We will be using (5.31) instead of (5.22), and (5.149) would still hold.

Unlike in Corollary 5.13, here we do not fix up d_L among the family of networks considered. Hence, following the recipe from Section 5.9, similar to (5.150), the term that would be different across the two networks is

$$\frac{d_L}{N\zeta^2} + \frac{1}{B}\left(\frac{13\gamma^2}{256\zeta}d_{L-1}d_L + \alpha_{L-1}d_{L-2}d_{L-1}d_L + \frac{\gamma}{4}\sum_{l=1}^{L-2}\alpha_l d_{l-1}d_l d_{l+1}\right) \tag{5.171}$$

Now all the networks that satisfy the following condition, will have the same (5.171), and hence the decay bound would be the same i.e., they all are equivalent according to Definition 5.12.

$$\begin{aligned} &\zeta\alpha_l d_{l-1}d_l d_{l+1} = \frac{1}{\Psi_l} \qquad \alpha_{L-1}d_{L-2}d_{L-1}d_L = \frac{1}{\Psi_{L-1}} \\ &\frac{d_{L-1}d_L}{\zeta} = \frac{1}{\Psi_L} \qquad \frac{d_L}{\zeta^2} = \frac{1}{\Psi_f} \qquad \text{for a given } \Psi_1,\ldots,\Psi_L \text{ and } \Psi_f \end{aligned} \tag{5.172}$$

Proof of Corollary 5.20

The proof follows exactly the same set of steps of comparing the bound for $\mathcal{D}_S$ and $\mathcal{D}_t$, as in (5.152)–(5.157) from the proof for Corollary 5.14. The only change is the presence of the dropout rate ζ, which then implies that, instead of the bound from (5.22) for constructing (5.152) we use (5.31). This then gives the following, similar to (5.152),

$$\begin{aligned} &\frac{1}{e_\gamma^m}\left(\frac{d_L}{N\zeta^2} + \frac{1}{B}\left(\frac{13d_s d_L\gamma^2}{256\zeta} + \frac{\alpha\gamma w}{4}(d_s^2 d_L + \zeta d_s^2 d_0) + \frac{\alpha\gamma\zeta w}{4}(L_s-3)d_s^3\right)\right) \\ &\quad \geqslant \frac{1}{e_\gamma^m}\left(\frac{d_L}{N\zeta^2} + \frac{1}{B}\left(\frac{13d_t d_L\gamma^2}{256\zeta} + \frac{\alpha\gamma w}{4}(d_t^2 d_L + \zeta d_t^2 d_0) + \frac{\alpha\gamma\zeta w}{4}(L_t-3)d_t^3\right)\right) \\ &\qquad \text{where} \quad e_\gamma^m = \gamma - \frac{13}{16}\gamma^2 \end{aligned} \tag{5.173}$$

Following the same recipe from (5.153)–(5.157), we will arrive at

$$L_t - 3 \leqslant (L_s-3)(1+\delta_{st})^3 + \frac{1}{\alpha\zeta w d_t^2}\left(\frac{13d_L\gamma}{64\zeta}\delta_{st} + \alpha w\delta_{st}(2+\delta_{st})(d_L+\zeta d_0)d_t\right) \tag{5.174}$$

Since $\delta_{st} > 0$, any L_t that satisfies the following set of inequalities will follow the above one (where we used the definitions of e_l^ms and the fact that $d_t = \frac{d_s}{1+\delta_{st}}$).

$$\begin{aligned} L_t &\leqslant L_s + 3\delta_{st}(L_s - 3) + \delta_{st}(1+\delta_{st})^2 \left(\frac{\frac{13 d_s d_L \gamma^2}{256}}{\alpha\zeta^2 \frac{\gamma w}{4} d_s^3} + \frac{\frac{\gamma w}{4} d_s^2 d_L}{\zeta \frac{\gamma w}{4} d_s^3} + \frac{\frac{\gamma w}{4} d_s^2 d_0}{\frac{\gamma w}{4} d_s^3} \right) \\ &= L_s + 3\delta_{st}(L_s - 3) + \delta_{st}(1+\delta_{st})^2 \left(\frac{e_L^m}{\alpha\zeta^2 e_2^m} + \frac{e_{L-1}^m}{\zeta e_2^m} + \frac{e_1^m}{e_2^m} \right) \end{aligned} \tag{5.175}$$

Chapter 6

Hierarchical modeling for Outcome and Enricher Design

6.1 Introduction

In Chapters 3, 4 and 5 of this report, the focus has been on hypothesis testing and supervised learning algorithms for designing efficient outcomes and/or enrichers in clinical trials. Implicitly both these approaches *require* validated constructs about the inputs (e.g., imaging features, disease markers, cognitive scores) and the outputs (e.g., disease status, predictive decline). In other words, *we need to know what we are looking for* – the methods are not exploratory. To see this, observe that recruiting younger middle-aged adults for clinical trials, would pose a more natural question to the biologists, neuro-scientists – and the computational models used for – studying pathology of the disease. Are the hypotheses used to design computational outcomes and/or enrichers valid across all stages of AD? For instance, are the complex interactions of imaging and cognitive summaries "similar" in early MCI and AD stages, and if not, how does one account for such changes input-to-output mapping as we setup the trial. Hence, the success of computational methods in trial design necessitates algorithms that *explore* hypotheses in tandem with exploiting the ideas presented in the earlier chapters. The focus of this chapter will be algorithms where we will *explicitly* model non-trivial structures within the datasets. We make this more precise shortly after introducing two fundamental aspects that are critical to the proposed algorithms – matrix factorization and multiresolution analysis.

Matrix factorization lies at the heart of a spectrum of computational problems in a multitude of domains in statistics including machine learning, com-

puter vision, signal processing, bioinformatics etc. While the wide ranging and extensive use of factorization schemes within feature/instance selection (Hoyer, 2004; Koren et al., 2009), basis pursuit (Chen et al., 2001), structure from motion (Sturm and Triggs, 1996), recognition (Turk and Pentland, 1991) and segmentation (Cheriyadat and Radke, 2009) have been known, in the last decade, there is renewed interest in these ideas. The celebrated work on low rank matrix completion (Candès and Recht, 2009) has enabled deployments in a broad cross-section of vision problems from independent components analysis (Hyvärinen, 2013) to dimensionality reduction (Wright et al., 2009) and background estimation (Xu et al., 2013). Novel extensions based on Robust Principal Components Analysis (De la Torre and Black, 2001; Candès and Recht, 2009) are being developed each year. Such factorization techniques, starting from classical principal components, to non-negative and hierarchical matrix factorizations, have been rigorously studied both theoretically and towards efficient algorithms. Invariably, the decomposition imposes some form of low-rank, group sparse, or some specific block structure (Chandrasekaran et al., 2005) on the matrix. More recently, block low-rank (Savas et al., 2011; Si et al., 2014) and hierarchical matrix approximation (Lee et al., 2008) methods have been proposed.

In contrast to factorization methods, a distinct and rich body of work based on early work in signal processing is arguably even more extensively utilized in statistics and machine learning. This is multiresolution analysis (MRA). Specifically, wavelets (Rubinstein et al., 2010) and other related ideas like curvelets (Candes and Donoho, 2000), shearlets (Kutyniok et al., 2012)) that loosely fall under MRA based approaches drive an overwhelming majority of techniques within feature extraction (Manjunath and Ma, 1996) and representation learning (Rubinstein et al., 2010). Also, Wavelets remain the "go to" tool for image denoising, compression, inpainting, shape analysis and other applications in video processing (Meyer, 1993). SIFT features can be thought of as a special case of the so-called scattering transform (using theory of wavelets) (Bruna and Mallat, 2013). Remarkably, the "network" perspective of scattering transform at least partly explains the invariances being identified by deep representations, further expanding the scope of multiresolution approaches informing vision algorithms. Although such multiresolution methods on continuous spaces have been around for a while, it is only recently that constructing wavelet bases on discrete spaces has been receiving much attention, starting from the seminal work on diffusion wavelets (Coifman and Maggioni, 2006).

The foregoing discussion raises the question of whether there are any interesting bridges between factorization and wavelets. Multiresolution operators, in general, correspond to sequence of decompositions, and hence it is reasonable to expect some fundamental relationship between the two techniques. This line of enquiry has recently been studied for the most common "discrete" object encountered in vision – graphs. Starting from diffusion wavelets, several authors have investigated tree-like decompositions on matrices (Lee et al., 2008), and methods for organizing them using wavelets (Gavish and Coifman, 2012). While the topic is still nascent (but evolving), these non-trivial results suggest that the confluence of these seemingly distinct topics potentially holds much promise for vision problems (Hwa Kim et al., 2016). Our focus is to study this interface between wavelets and factorization, and demonstrate the among immediate set of problems that can potentially benefit includes selection, ranking and exploring feature-wise (and subject-wise) interactions for designing outcomes and/or enricher in clinical trials.

Before we present the overview of this contribution, there is an alternate motivation for bridging matrix factorization methods and multiresolution analysis from the perspective of understanding (and interpreting) relationships between learned representations. Unlike exploring feature (or subject) interactions for trial design, here we are interested in class-wise or category-wise interactions. The motivation for this problem of interpreting learned representations, and the solution we provide, corresponds to a wide-range of questions at the intersection of interpretable learning models and "black-box" machine learning.

Decoding learned representations

The ability of a feature to succinctly represent the presence of an object is, at least, in part, governed by the relationship of the learned representations across multiple classes/categories. Such objects can be, for instance, tumors on brain scans, mutations in gene sequences, daily-use objects like phone, car etc. on natural images. Cross-covariate contextual dependencies have been shown to improve the performance, for instance, in medical applications (Ithapu et al., 2015b), and object tracking and recognition in vision (Zhang et al., 2012; Su and Jurie, 2012) – a motivating aspect of adversarial learning (Lowd and Meek, 2005). Since humans typically have good perceptive ability to perform these tasks, it then poses an interesting question – *Do the semantic relationships learned*

by the deep representations associate with those seen by humans? Or in other words, can the relationships between different tasks (or categories) inferred by the highest layer representations 'drive' the design of the network itself (e.g., to add an additional layer or to remove an existing one)? For instance, can such models infer that cats are closer to dogs than they are to bears; or that bread goes well with butter/cream rather than, say, salsa. Invariably, addressing these questions amounts to learning hierarchical and categorical relationships in the class-covariance of hidden representations. Using classical techniques may not easily reveal interesting, human-relateable trends as recently shown by (Peterson et al., 2016). Note that the problem here is not about constructing learning models that *model* the semantics, like those using statistical relational learning (Mooney, 1999) or rule-based methods (DeJong and Spears, 1990; Lakkaraju and Rudin, 2017). Instead, we are asking whether one can *decode* the semantics of representations from an *arbitrary* (trained) network.

Related Work: The problem of understanding the landscape of representations learned by machine learning models is not entirely new, and interpreting learning models has a long history (Letham et al., 2015; Lou et al., 2012; Lakkaraju and Leskovec, 2016). For instance, some approaches construct the learning model to be parsable to begin with, like decision sets (Wang et al., 2015) or decision lists (Letham et al., 2015). Alternatively, the learned features are inverted appropriately to visualize what the learning model sees (e.g. HOGgles (Vondrick et al., 2013)). Several recent studies have approached the problem of *interpreting of deep representations* from alternate view-points (Lipton, 2016). The seminal papers on deep networks (Erhan et al., 2010c,b; Yosinski et al., 2014) argue that, intuitively and empirically, higher layer representations learn abstract relationships between objects in a given image. However, it is unclear whether we can "define" what these abstract relationships (that the network should learn) are supposed to be. In (Simonyan et al., 2013; Yosinski et al., 2015; Zeiler and Fergus, 2014), the authors visualize the representation classes using supervised learning methods, and extra semantic information is provided during training time to improve interpretability (Dong et al., 2017). (Simonyan et al., 2013; Dosovitskiy and Brox, 2015) have addressed similar aspects for deep representations by visualizing image classification and detection models, and there is recent interest in designing tools for visualizing what the network perceives when predicting a test label (Yosinski et al., 2015). As shown in (Mordvintsev et al., 2015), the contextual images that a deep network (even with good detection power) desires to see may not even correspond to real-world

scenarios.

Overview

The contribution of this work is a general framework, and an exploratory tool, that models the hierarchical relationships between several classes (or categories, tasks). The feature representations may come directly from data, like MRI or PET images, or may be learned using for instance state-of-the-art deep networks. To concretize the arguments from above, consider a representative example in machine learning where a matrix factorization approach may be deployed. Figure 6.1 shows a set of covariance matrices computed from the representations learned by VGG-S (Chatfield et al., 2014) (on some ImageNet classes (Russakovsky et al., 2015)), ADNI brain imaging ROIs and WRAP cognitive scores respectively. As a first line of exploration, we may be interested in characterizing the apparent parsimonious "structure" seen in these matrices. We can easily verify that invoking the de facto constructs like sparsity, low-rank or a decaying eigen-spectrum cannot account for the "block" or cluster-like structures inherent in this data. Such block-structured kernels were the original motivation for block low-rank and hierarchical factorizations (Savas et al., 2011; Chandrasekaran et al., 2005; Liang et al., 2014; Zhang et al., 2013). However these existing methods are mainly heuristic, sensitive to the choice of several hyperparameters involved (like the choice of local patch size, the local-rank etc.), require making hard-partitioning of data into clusters, non-trivial to compute for large dimensions and mostly do not necessarily capture the global and local structure equally well.

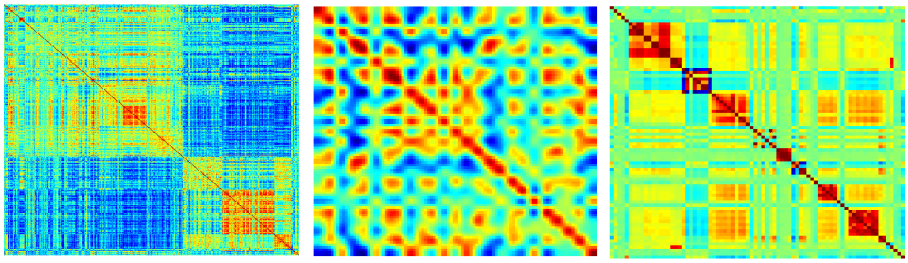


Figure 6.1: Example covariance matrices

A multiresolution scheme is much more natural — in fact, ideal — if one can decompose the matrix in a way that the blocks automatically 'reveal' themselves

at multiple resolutions. Conceptually, this amounts to a sequential factorization while accounting for the fact that each level of this hierarchy must correspond to approximating some non-trivial structure in the matrix. A recent result introduces precisely such a *multiresolution matrix factorization* (MMF) algorithm for symmetric matrices (Kondor et al., 2014). Consider a symmetric matrix $\mathbf{C} \in \mathbb{R}^{m \times m}$. PCA decomposes $\mathbf{C}$ as $\mathbf{Q}^T \mathbf{R} \mathbf{Q}$ where $\mathbf{Q}$ is an orthogonal matrix, which, in general, is dense. On the other hand, sparse PCA (sPCA) (Zou et al., 2006) imposes sparsity on the columns of $\mathbf{Q}$, allowing for fewer dimensions to interact that may not capture global patterns. The factorization resulting from such individual low-rank decompositions cannot capture hierarchical relationships among data dimensions. Instead, MMF applies a sequence of carefully chosen sparse rotations $\mathbf{Q}^1, \mathbf{Q}^2, \ldots \mathbf{Q}^L$ to factorize $\mathbf{C}$ in the form

$$\mathbf{C} = (\mathbf{Q}^1)^T (\mathbf{Q}^2)^T \ldots (\mathbf{Q}^L)^T \mathbf{R} \mathbf{Q}^L \ldots \mathbf{Q}^2 \mathbf{Q}^1,$$

thereby uncovering soft hierarchical organization of different rows/columns of $\mathbf{C}$. Typically the $\mathbf{Q}^\ell$s are sparse k^{th}-order rotations (orthogonal matrices that are the identity except for at most k of their rows/columns), leading to a hierarchical tree-like matrix organization. MMF was shown to be an efficient compression tool (Teneva et al., 2016) and a preconditioner (Kondor et al., 2014). Randomized heuristics have been proposed to handle large matrices (Kondor et al., 2015). Nevertheless, factorization involves searching a combinatorial space of row/column indices, which restricts the order of the rotations to be small (typically, $\leqslant 3$). Not allowing higher order rotations restricts the richness of the allowable block structure, resulting in a hierarchical decomposition that is "too localized" to be sensible or informative (reverting back to the issues with sPCA and other block low-rank approximations).

Until recently randomized heuristics have been proposed to handle large k and large matrices (Kondor et al., 2015). The main contribution of this work is a novel algorithm for estimating MMF by observing that many of the computations involved in the exhaustive procedures (Teneva et al., 2016; Kondor et al., 2015) may be by-passed or approximated to a high fidelity. Note that a fundamental property of MMF is the sequential composition of rotations. In developing the procedure, we exploit the fact that the factorization can be parameterized in terms of an *MMF graph* defined on a sequence of higher-order rotations. Unlike alternate batch-wise approaches (Teneva et al., 2016), we start with a small, randomly chosen block of $\mathbf{C}$, and gradually 'insert' new rows into

the factorization – hence we refer to this as an incremental MMF. We show that this insertion procedure manipulates the topology of the MMF graph, thereby providing an efficient algorithm for constructing higher order MMFs. The proposed ideas are summarized in Ithapu et al. (2017a). Our contributions are: **(A)** We present a fast and efficient incremental procedure for constructing higher order (large k) MMFs on large dense matrices; **(B)** Using the new algorithm we generalize statistical leverage scores – *MMF scores*, for ranking and selection of features/instances; outlier detection etc. **(C)** Using the output structure of incremental MMF – *MMF graph*, we visualize the semantics of categorical relationships inferred by learning models, specifically, neural networks, and, in turn, present some exploratory tools to adapt and modify the architectures. **(D)** Finally, we show that MMF scores and graphs build upon the proposed approaches in earlier chapters for designing outcomes and enrichers in clinical trials.

6.2 Multiresolution Matrix Factorization

We begin with some notation. Recall that matrices are bold upper case, vectors are bold lower case and scalars are lower case. $[m] := \{1, \ldots, m\}$ for any $m \in \mathbb{N}$. Given a matrix $\mathbf{C} \in \mathbb{R}^{m \times m}$ and two set of indices $\mathcal{S}_1 = \{r_1, \ldots r_k\}$ and $\mathcal{S}_2 = \{c_1, \ldots c_p\}$, $\mathbf{C}_{\mathcal{S}_1,\mathcal{S}_2}$ will denote the block of $\mathbf{C}$ cut out by the rows $\mathcal{S}_1$ and columns $\mathcal{S}_2$. $\mathbf{C}_{:,i}$ is the i^{th} column of $\mathbf{C}$. $\mathbf{I}_m$ is the m-dimensional identity. $SO(m)$ is the group of m dimensional orthogonal matrices with unit determinant. $\mathcal{R}^m_{\mathcal{S}}$ is the set of m-dimensional symmetric matrices which are diagonal except for their $\mathcal{S} \times \mathcal{S}$ block – these are also referred to as $\mathcal{S}$–core-diagonal matrices. Multiresolution matrix factorization (MMF), introduced in (Kondor et al., 2014, 2015), retains the locality properties of sPCA while also capturing the global interactions provided by the many variants of PCA, by applying not one, but multiple *sparse* rotation matrices to $\mathbf{C}$ in sequence. We have the following.

Definition 6.1. *Given an appropriate class $\mathcal{O} \subseteq SO(m)$ of sparse rotation matrices, a depth parameter $L \in \mathbb{N}$ and a sequence of integers $m = d_0 \geqslant d_1 \geqslant \ldots \geqslant d_L \geqslant 1$, the **multi-resolution matrix factorization (MMF)** of a symmetric matrix $\mathbf{C} \in \mathbb{R}^{m \times m}$ is a factorization of the form*

$$\mathcal{M}(\mathbf{C}) := \overline{\mathbf{Q}}^T \mathbf{R} \overline{\mathbf{Q}} \quad \textit{with} \quad \overline{\mathbf{Q}} = \mathbf{Q}^L \ldots \mathbf{Q}^2 \mathbf{Q}^1, \tag{6.1}$$

where $\mathbf{Q}^\ell \in \mathcal{O}$ and $\mathbf{Q}^\ell_{[m]\setminus\mathcal{S}_{\ell-1},[m]\setminus\mathcal{S}_{\ell-1}} = \mathbf{I}_{m-d_\ell}$ for some nested sequence of sets $[m] = \mathcal{S}_0 \supseteq \mathcal{S}_1 \supseteq \ldots \supseteq \mathcal{S}_L$ with $|\mathcal{S}_\ell| = d_\ell$ and $\mathbf{R} \in \mathcal{R}^m_{\mathcal{S}_L}$.

Before we discuss the nature of this decomposition, we will briefly discuss MMF from the context of modeling complex, hierarchical, and parsimonious interactions in symmetric matrices.

Understanding MMF

The factorization in Definition 6.1 can be interpreted as a *bridge* between two different schools of modeling relational data structures i.e., similarity, kernel or covariance matrices – as discussed earlier in Section 6.1. These include methods that model symmetric matrices via global factorization (like PCA) and those that model local interactions hierarchically (like treelets (Lee et al., 2008)). MMF bridges them by modeling *global interactions as sequential compositions of local correlations*. Such a sequential composition – which can be seen clearly from (6.1) – naturally forms a hierarchy. Such interesting hierarchies can be seen in the example Figure 6.1. As suggested in Section 6.1, existing approaches for pursuing such a modeling of matrices have been "heuristic" (Savas et al., 2011; Si et al., 2014), or, in many cases, have restricted the interaction structure (hierarchy) to a tree (Rajani et al., 2015). Observe that we are concerned about three attributes of matrix modeling here – all of which are directly motivated to *bypass* any such heuristics.

1. *Localization:* Subsets of entities (here, the rows/columns of the matrix) need to be decorrelated so as to impose an implicit ranking/hierarchy among the strengths of relationships.
2. *Multi-scale:* While localization takes into account the precise subsets of entities that are interacting, a multi-scale transformation is required to impose higher-order relationships (subsets of entities belonging to different frequencies).
3. *Data-driven:* The procedure should be governed by as few hyper-parameters as possible, while some physical/statistical property attributed to each such hyperparameter.

MMF satisfies all these three criteria, making it a powerful technique to model symmetric matrices. First observe that factorization has L levels where $\mathbf{Q}^\ell \forall \ell$ are sparse rotation matrices. $\mathcal{S}_{\ell-1}$ is referred to as the 'active set' at the ℓ^{th} level, since $\mathbf{Q}^\ell$ is identity outside $[m] \setminus \mathcal{S}_{\ell-1}$. The nesting of the $\mathcal{S}_\ell$s implies that after applying $\mathbf{Q}^\ell$ at some level ℓ, $\mathcal{S}_{\ell-1} \setminus \mathcal{S}_\ell$ rows/columns are removed from the active set, and are not operated on subsequently. This active set trimming is done at all L levels, leading to a nested subspace interpretation for the sequence

of compressions $\mathbf{C}^\ell = \mathbf{Q}^\ell \mathbf{C}^{\ell-1} (\mathbf{Q}^\ell)^\mathsf{T}$ ($\mathbf{C}^0 = \mathbf{C}$ and $\mathbf{R} = \mathbf{C}^\mathrm{L}$).

This sequential nesting and compression is the 'core' aspect of MMF that makes it multi-scale and data-driven while computing localized decorrelation. Let us now visualize this same set of operations before discussing the precise procedure for computing the unknown $\mathbf{Q}^\ell$s for a given $\mathbf{C}$ and L. The nesting is shown in Figure 6.2. The given symmetric matrix $\mathbf{C}^0$ is left and right multiplied with a rotation matrix such that, only a fraction of the rows/columns of the rotation matrix are non-identity (the red blocks in Figure 6.2(a)). The active sets $\mathcal{S}_1, \mathcal{S}_2, \ldots$ are nested, and as shown in Figure 6.2(b), the rows/columns indexed by green are *allowed* to be rotated at first level, but not in the second level. The process repeats for a fixed number of levels (see Figure 6.2(c)). In fact, the authors of (Kondor et al., 2014) have shown that, for a general class of symmetric matrices, MMF from Definition 6.1 entails a Mallat style multiresolution analysis (MRA) (Mallat, 1989). Observe that depending on the choice of $\mathbf{Q}^\ell$, only a few dimensions of $\mathbf{C}^{\ell-1}$ are forced to interact, and so the composition of rotations is hypothesized to extract subtle or softer notions of structure in $\mathbf{C}$. Since multiresolution is represented as matrix factorization here (see (6.1)), the $\mathcal{S}_{\ell-1} \setminus \mathcal{S}_\ell$ columns of $\overline{\mathbf{Q}}$ correspond to "wavelets".

Computing the factorization

While $\mathrm{d}_1, \mathrm{d}_2, \ldots$ (which correspond to the sizes of the active sets $\mathcal{S}_\ell$) can be any monotonically decreasing sequence, we restrict ourselves to the simplest case of $\mathrm{d}_\ell = \mathrm{m} - \ell$. Within this setting, the number of levels L is at most $\mathrm{m} - \mathrm{k} + 1$, and each level contributes a single wavelet since $|\mathcal{S}_{\ell-1} \setminus \mathcal{S}_\ell| = 1 \forall \ell$ (see discussion above and Figure 6.2). Note that the algorithmic and optimization aspects follow through for any other choices of d_ℓs as well, however this simple case of arithmetic decay of d_ℓs suffices for our presentation. The matrix factorization of (6.1) then reduces to determining the $\mathbf{Q}^\ell$ rotations and the residual $\mathbf{R}$, which is usually done by minimizing the squared Frobenius norm error

$$\min_{\mathbf{Q}^\ell \in \mathcal{O}, \mathbf{R} \in \mathcal{R}^{\mathrm{m}}_{\mathrm{s}_\mathrm{L}}} \|\mathbf{C} - \mathcal{M}(\mathbf{C})\|^2_{\mathrm{Frob}}. \tag{6.2}$$

This is clearly a non-convex optimization problem because of the nesting (product compositions) structure in $\mathcal{M}(\mathbf{C})$ from (6.1). However, the above objective can be decomposed as a sum of contributions from each of the L different levels (Proposition 1, (Kondor et al., 2014)), which suggests computing the factorization in a greedy manner as $\mathbf{C} = \mathbf{C}^0 \mapsto \mathbf{C}^1 \mapsto \mathbf{C}^2 \mapsto \ldots \mapsto \mathbf{R}$. This error decomposition is what drives much of the intuition behind our algorithms.

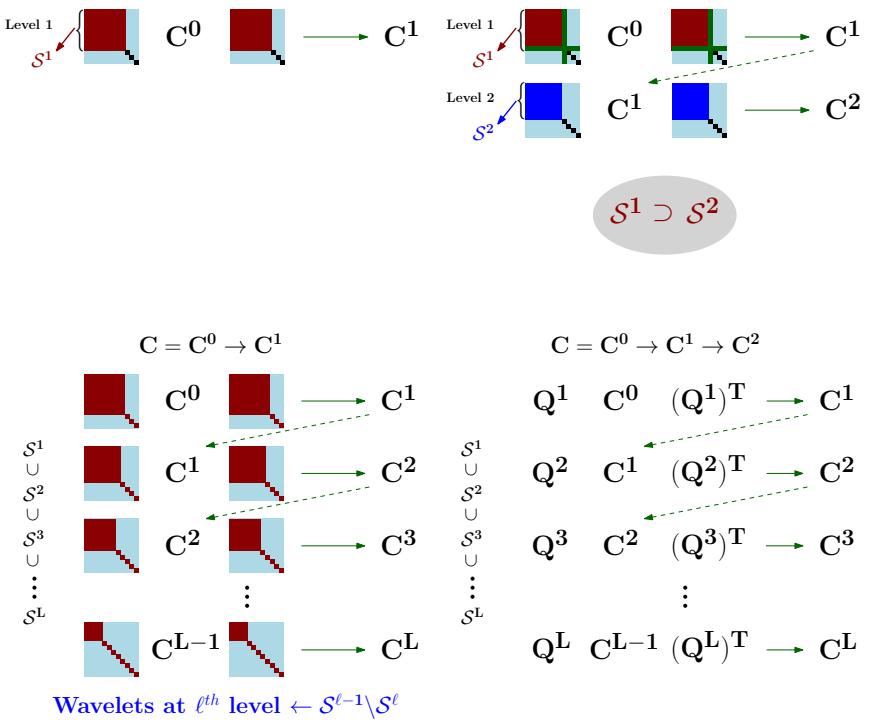


Figure 6.2: Visualizing the sequence of operations performed by MMF on $\mathbf{C}$.

Recall that after $\ell - 1$ levels, $\mathbf{C}^{\ell-1}$ is the compression and $\mathcal{S}_{\ell-1}$ is the new active set. In the simplest case of $\mathcal{O}$ being the class of so-called k–point Givens rotations (i.e., rotations which affect at most k coordinates) with $\mathcal{S}_0 = [m]$ and $d_\ell = m - \ell$, at level ℓ the algorithm needs to determine three things: **(a)** the k–tuple t^ℓ of rows/columns involved in the rotation, **(b)** the nontrivial part $\mathbf{O} := \mathbf{Q}^\ell_{t^\ell,t^\ell}$ of the rotation matrix, and **(c)** s^ℓ, the index of the row/column that is subsequently designated a wavelet and removed from the active set. Without loss of generality, let s^ℓ be the last element of t^ℓ. Then the contribution of level ℓ to the squared Frobenius norm error (6.2) is (Proposition 2, (Kondor et al., 2014))

$$\begin{aligned} \mathcal{E}(\mathbf{C}^{\ell-1}; \mathbf{O}; t^\ell, s^\ell) &= 2\sum_{i=1}^{k-1} [\mathbf{O}\mathbf{C}^{\ell-1}_{t^\ell,t^\ell}\mathbf{O}^T]^2_{k,i} \\ &+ 2[\mathbf{O}\mathbf{B}\mathbf{B}^T\mathbf{O}^T]_{k,k} \quad \text{where} \quad \mathbf{B} = \mathbf{C}^{\ell-1}_{t^\ell,\mathcal{S}_{\ell-1}\backslash t^\ell}, \end{aligned} \tag{6.3}$$

Observe that in the definition of $\mathbf{B}$, t^ℓ is treated as a set. One then needs to

choose the best possible t^ℓ and the corresponding rotation $\mathbf{O}$ so that the above error is minimized. In general, these is no guarantee that the error will be zero for a choice of t^ℓ and $\mathbf{O}$, thereby we do not have an analytic solution for (6.3) (unlike, for instance, with eigenbasis computation). Hence, the factorization then works by minimizing this quantity in a *greedy* fashion, i.e.,

$$\begin{aligned} \mathbf{Q}^\ell, t^\ell, s^\ell &\leftarrow \underset{\mathbf{O},t,s}{\arg\min}\, \mathcal{E}(\mathbf{C}^{\ell-1}; \mathbf{O}; t, s) \\ \mathcal{S}_\ell &\leftarrow \mathcal{S}_{\ell-1} \setminus s^\ell \quad ; \quad \mathbf{C}^\ell = \mathbf{Q}^\ell \mathbf{C}^{\ell-1} (\mathbf{Q}^\ell)^T. \end{aligned} \tag{6.4}$$

Batch MMF: The greedy approach that explicitly minimizes the error criterion in (6.3) for all the levels ℓ is referred to as the batch MMF. This works by first fixing the dictionary of k^{th}-order rotations (i.e., $\mathcal{O}$), for instance, using properties of QR decomposition (Mezzadri, 2006). At every level $\ell = 1, 2, \ldots$, the algorithm exhaustively searches over all possible k-tuples from $\mathcal{S}_{\ell-1}$ (the current active set) to chose the best t^ℓ. If the dictionary is large enough, the exhaustive procedure would lead to the smallest possible decomposition error (see (6.2)). However, note that the selection of the best k-tuples here is clearly combinatorial, with an overall complexity of $\mathcal{O}(n^k)$ (Kondor et al., 2015; Teneva et al., 2016). Hence, the exact MMF computation, i.e., explicitly minimizing (6.2), is very expensive even for $k = 3$ or 4. This is one of the primary motivations for the proposed improved factorization in the next section. The batch MMF is summarized in Algorithm 5.

Other Variants – Eigen MMF and Random MMF: The are two alternatives that avoid this exhaustive search. Since $\mathbf{Q}^\ell$'s job is to diagonalize some k rows/-columns (see Definition 6.1), one can simply pick the relevant $k \times k$ block of $\mathbf{C}^\ell$ and compute the best $\mathbf{O}$ (for a given t^ℓ). Hence the first alternative called Eigen MMF is to bypass the search over $\mathcal{O}$ (in (6.4)), and simply use the eigenvectors of $\mathbf{C}^\ell_{t^\ell, t^\ell}$ for some tuple t^ℓ. Nevertheless, the search over $\mathcal{S}_{\ell-1}$ for t^ℓ still makes this approximation costly. Instead, in the second alternative, the k-tuple selection may be approximated while keeping the exhaustive search over $\mathcal{O}$ intact (Kondor et al., 2015). Since diagonalization effectively nullifies correlated dimensions, the best k-tuple can be the k rows/columns that are maximally correlated. This is done by choosing some $s_1 \sim \mathcal{S}_{\ell-1}$ (from the current active set), and picking the rest by

$$s_2, \ldots, s_k \leftarrow \underset{s_i \sim \mathcal{S}_{\ell-1} \setminus s_1}{\arg\min} \sum_{i=2}^{k} \frac{(\mathbf{C}^{\ell-1}_{:,s_1})^T \mathbf{C}^{\ell-1}_{:,s_i}}{\|\mathbf{C}^{\ell-1}_{:,s_1}\| \|\mathbf{C}^{\ell-1}_{:,s_i}\|} \tag{6.5}$$

This second heuristic (which is related to (6.7) from Section 6.3) has been shown to be robust (Kondor et al., 2015), however, for large k it is guaranteed to miss many combinations of k-tuples that are vital to the quality of the factorization. This aspect directly follows from sampling in high dimensions with finite computational load – and hence would be referred to as Random MMF. For brevity, we will not discuss technical results which show that the above greedy error minimization procedure computes s^ℓs that correspond to Mallat-style translation-and-dilation wavelets i.e., the sequence $\mathbf{C}^0 \mapsto \mathbf{C}^1 \mapsto \mathbf{C}^2 \mapsto \dots$ is in fact MRA. For these details, we direct the reader to (Kondor et al., 2014).

Algorithm 5 BATCH MMF($\mathbf{C}, k, \mathcal{O}$)

Output: $\mathcal{M}(\mathbf{C})$
  $L = m - k + 1$, $\mathcal{S}_0 = [m]$
  for $\ell \in \{1, \dots, L\}$ **do**
    $\mathbf{Q}^\ell, t^\ell, s^\ell \leftarrow \arg\min_{\mathbf{O},t,s} \mathcal{E}(\mathbf{C}^{\ell-1}; \mathbf{O}; t, s)$ from (6.3)
    $\mathcal{S}_\ell \leftarrow \mathcal{S}_{\ell-1} \setminus s^\ell$
    $\mathbf{C}^\ell = \mathbf{Q}^\ell \mathbf{C}^{\ell-1} (\mathbf{Q}^\ell)^\mathsf{T}$
  end for

6.3 Incremental MMF

The necessity for improving the factorization directly comes from (6.3) and (6.4). Observe that solving (6.2) amounts to estimating the L different k-tuples $t^1, \dots, t^L$ sequentially, and the fact that the exhaustive procedure is combinatorial and expensive.Higher order MMFs (with large k) are desirable for allowing arbitrary interactions among dimensions, as discussed in Section 6.1 – we will also provide experimental evidence for these claims later in Section 6.6. We will now present a faster algorithm for computing the factorization. It exploits some interesting properties of the factorization error and other redundancies in k-tuple computation. The core of our proposal is the following setup.

Overview

Let $\tilde{\mathbf{C}} \in \mathbb{R}^{(m+1)\times(m+1)}$ be the extension of $\mathbf{C}$ by a *single* new column $\mathbf{w} = [\mathbf{u}^\mathsf{T}, \nu]^\mathsf{T}$, which manipulates $\mathbf{C}$ as:

$$\tilde{\mathbf{C}} = \left[\begin{array}{c|c} \mathbf{C} & \mathbf{u} \\ \hline \mathbf{u}^\mathsf{T} & \nu \end{array}\right] . \tag{6.6}$$

The goal is to compute $\mathcal{M}(\tilde{\mathbf{C}})$. Since $\mathbf{C}$ and $\tilde{\mathbf{C}}$ share all but one row/column (see (6.6)), if we have access to $\mathcal{M}(\mathbf{C})$, one should, in principle, be able to modify $\mathbf{C}$'s underlying sequence of rotations to construct $\mathcal{M}(\tilde{\mathbf{C}})$. This avoids having to recompute everything for $\tilde{\mathbf{C}}$ from scratch, i.e., performing the greedy decompositions from (6.4) on the entire $\tilde{\mathbf{C}}$. The hypothesis for manipulating $\mathcal{M}(\mathbf{C})$ to compute $\mathcal{M}(\tilde{\mathbf{C}})$ comes from the precise computations involved in the factorization. Recall (6.3) and the discussion leading up to the expression. At level $\ell+1$, the factorization picks the 'best' candidate rows/columns from $\mathbf{C}^\ell$ that correlate the most with each other, so that the resulting diagonalization induces the smallest possible off-diagonal error over the rest of the active set. The expression in (6.3) encompasses the error in this diagonalization. The components contributing towards this error are driven by the inner products $(\mathbf{C}^\ell_{:,i})^\mathsf{T}\mathbf{C}^\ell_{:,j}$ for some columns i and j. In some sense, the largest such correlated rows/columns get picked up, and adding one new entry to $\mathbf{C}^\ell_{:,i}$ may not change the "range" of these correlations. Extending this intuition across all levels, we argue that

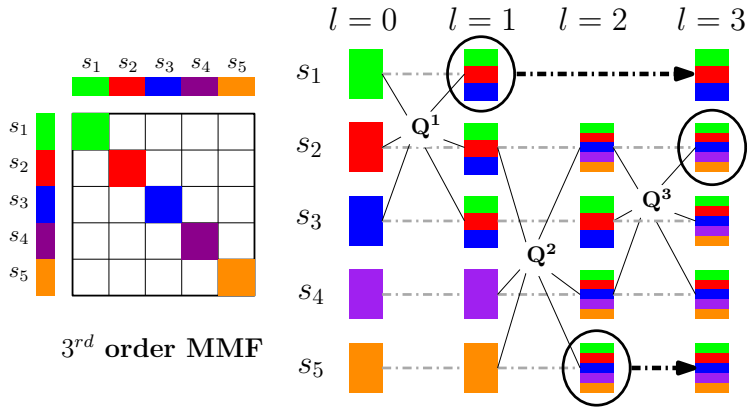
$$\operatorname*{argmax}_{i,j} \tilde{\mathbf{C}}^\mathsf{T}_{:,i}\tilde{\mathbf{C}}_{:,j} \approx \operatorname*{argmax}_{i,j} \mathbf{C}^\mathsf{T}_{:,i}\mathbf{C}_{:,j} \tag{6.7}$$

Hence, the k-tuples computed from $\mathbf{C}$'s factorization are reasonably good candidates even after introducing $\mathbf{w}$. To better formalize this idea, and in the process present our algorithm, we parameterize the output structure of $\mathcal{M}(\mathbf{C})$ in terms of the sequence of rotations and the wavelets.

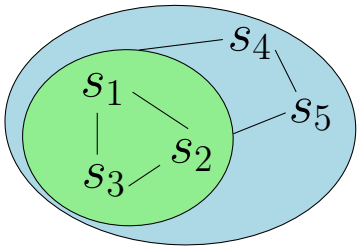
The graph structure of $\mathcal{M}(\mathbf{C})$

If one has access to the sequence of k-tuples $t^1, \ldots, t^L$ involved in the rotations and the corresponding wavelet indices $(s^1, \ldots, s^L)$, then the factorization is straightforward to compute i.e., there is no greedy search anymore. Recall that by definition $s^\ell \in t^\ell$ and $s^\ell \notin \mathcal{S}_\ell$ (see (6.4)). To that end, for a given $\mathcal{O}$ and L, $\mathcal{M}(\mathbf{C})$ can be 'equivalently' represented using a depth L *MMF graph* $\mathcal{G}(\mathbf{C})$. Each level of this graph shows the k-tuple t^ℓ involved in the rotation, and the corresponding wavelet s^ℓ i.e., $\mathcal{G}(\mathbf{C}) := \{t^\ell, s^\ell\}_1^L$. Interpreting the factorization in this way is notationally convenient for presenting the algorithm. More importantly, such an interpretation is central for visualizing hierarchical dependencies among dimensions of $\mathbf{C}$, and will be discussed in detail in Section 6.6. An example of such a 3^{rd} order MMF graph constructed from a 5×5 matrix is shown in Figure 6.3. The rows/columns are color coded for better

visualization. At level $\ell = 1$, s_1, s_2 and s_3 are diagonalized while designating the rotated s_1 as the wavelet. This process repeats for $\ell = 2$ and 3. As shown by the color-coding of different compositions, MMF gradually teases out *higher-order* correlations that can only be revealed after composing the rows/columns at one or more scales (levels here). Figure 6.3(b) shows the visualization of



(a) **Example Construction**



(b) **MMF graph Visualization**

Figure 6.3: An example 5×5 matrix, and its 3^{rd} order MMF graph.

the resulting MMF graph – each ellipse is a level and the black lines simply indicate interactions. To avoid clutter as ℓ increases, lines are only shown from higher level categories (s_4 and s_5 here) to lower level ellipses (and not s_1, s_2 and s_3). For notational convenience, we denote the MMF graphs of $\mathbf{C}$ and $\tilde{\mathbf{C}}$ as $\mathcal{G} := \{t^\ell, s^\ell\}_1^L$ and $\tilde{\mathcal{G}} := \{\tilde{t}^\ell, \tilde{s}^\ell\}_1^{L+1}$. Recall that $\tilde{\mathcal{G}}$ will have one more level than $\mathcal{G}$ since the row/column $\mathbf{w}$, indexed $m+1$ in $\tilde{\mathbf{C}}$, is being added (see (6.6)). The goal is to estimate $\tilde{\mathcal{G}}$ without recomputing all the k-tuples using the greedy procedure from (6.4). This translates to *inserting* the new index $m+1$ into the t^ℓs and modifying s^ℓs accordingly. Following the discussion from Section 6.3,

incremental MMF argues that inserting this one new element into the graph will not result in global changes in its topology. Clearly, in the pathological case, $\mathcal{G}$ may change arbitrarily, but as argued earlier (see discussion about (6.7)) the chance of this happening for non-random matrices with reasonably large k is small. The core operation then is to compare the new k-tuples resulting from the addition of $\mathbf{w}$ to the best ones from $[m]^k$ provided via $\mathcal{G}$. If the newer k-tuple gives better error (see (6.3)), then it will *knock out* an existing k-tuple. This constructive insertion and knock-out procedure is the incremental MMF.

Inserting a new row/column

The basis for this incremental procedure is that one has access to $\mathcal{G}$ (i.e., the MMF on $\mathbf{C}$). We first present the algorithm assuming that this "initialization" is provided, and revisit this aspect shortly. The procedure starts by setting $\tilde{t}^\ell = t^\ell$ and $\tilde{s}^\ell = s^\ell$ for $\ell = 1, \ldots, L$. Let $\mathcal{I}$ be the set of elements (indices) that needs to be inserted into $\mathcal{G}$. At the start (the first level) $\mathcal{I} = \{m+1\}$ corresponding to $\mathbf{w}$. Let $\tilde{t}^1 = \{p_1, \ldots, p_k\}$. The new k-tuples that account for inserting entries of $\mathcal{I}$ are $\{m+1\} \cup t^1 \setminus p_i$ $(i = 1, \ldots, k)$. These new k candidates are the probable alternatives for the existing $\tilde{t}^1$. Once the best among these $k+1$ candidates is chosen, an existing p_i from $\tilde{t}^1$ may be knocked out. If $\tilde{s}^1$ gets knocked out, then $\mathcal{I} = \{\tilde{s}^1\}$ for future levels. This follows from MMF construction, where wavelets at ℓ^{th} level are not involved in later levels. Since $\tilde{s}^1$ is knocked out, it is the new inserting element according to $\mathcal{G}$. On the other hand, if one of the $k-1$ scaling functions is knocked out, $\mathcal{I}$ is not updated. This simple process is repeated sequentially from $\ell = 1$ to L. At $L+1$, there are no estimates for $\tilde{t}^{L+1}$ and $\tilde{s}^{L+1}$, and so, the procedure simply selects the best k-tuple from the remaining active set $\tilde{\mathcal{S}}_L$. Algorithm 6 summarizes this insertion and knock-out procedure.

Incremental MMF Algorithm

Observe that Algorithm 6 is for the setting from (6.6) where one extra row/column is added to a given MMF, and clearly, the incremental procedure can be repeated as more and more rows/columns are added. Algorithm 8 summarizes this incremental factorization for arbitrarily large and dense matrices. It has two components: an initialization on some randomly chosen small block (of size $\tilde{m} \times \tilde{m}$) of the entire matrix $\mathbf{C}$; followed by insertion of the remaining $m - \tilde{m}$ rows/columns using Algorithm 6 in a streaming fashion (similar to $\mathbf{w}$ from (6.6)). The initialization entails computing an MMF on this small block

Algorithm 6 INSERTROW($\mathbf{C}, \mathbf{w}, \{t^\ell, s^\ell\}_{\ell=1}^{L}$)

Output: $\{\tilde{t}^\ell, \tilde{s}^\ell\}_{\ell=1}^{L+1}$
$\tilde{\mathbf{C}}^0 \leftarrow \tilde{\mathbf{C}}$ as in (6.6)
$z^1 \leftarrow m+1$
for $\ell = 1$ to $L-1$ **do**
$\{\tilde{t}^\ell, \tilde{s}^\ell, z^{\ell+1}, \mathbf{Q}^\ell\} \leftarrow$ CHECKINSERT($\tilde{\mathbf{C}}^{\ell-1}; t^\ell, s^\ell, z^\ell$)
$\tilde{\mathbf{C}}^\ell = \mathbf{Q}^\ell \tilde{\mathbf{C}}^{\ell-1} (\mathbf{Q}^\ell)^\top$
end for
$\mathcal{T} \leftarrow$ GENERATETUPLES($[m+1] \setminus \cup_{\ell=1}^{L-1} \tilde{s}^\ell(\tilde{\mathbf{C}})$)
$\{\tilde{\mathbf{O}}, \tilde{t}^L, \tilde{s}^L\} \leftarrow \arg\min_{\mathbf{O}, t \in \mathcal{T}, s \in t} \mathcal{E}(\tilde{\mathbf{C}}^{L-1}; \mathbf{O}; t, s)$
$\mathbf{Q}^L = \mathbf{I}_{m+1}$, $\mathbf{Q}^L_{\tilde{t}^L, \tilde{t}^L} = \tilde{\mathbf{O}}$, $\tilde{\mathbf{C}}^L = \mathbf{Q}^L \tilde{\mathbf{C}}^{L-1} (\mathbf{Q}^L)^\top$

Algorithm 7 CHECKINSERT($\mathbf{A}, \hat{t}, \hat{s}, z$)

Output: $\tilde{t}$, $\tilde{s}$, z, $\mathbf{Q}$
$\mathcal{T} \leftarrow$ GENERATETUPLES($\hat{t}, z$)
$\{\tilde{\mathbf{O}}, \tilde{t}, \tilde{s}\} \leftarrow \arg\min_{\mathbf{O}, t \in \mathcal{T}, s \in t} \mathcal{E}(\mathbf{A}; \mathbf{O}; t, s)$
if $\tilde{s} \in z$ **then**
$z \leftarrow (z \cup \hat{s}) \setminus \tilde{s}$
end if
$\mathbf{Q} = \mathbf{I}_{m+1}$, $\mathbf{Q}_{\tilde{t}, \tilde{t}} = \tilde{\mathbf{O}}$

($\tilde{m} \geqslant k$) using Algorithm 6.2. Depending on $\tilde{m}$ and the available computational resources at hand, either batch MMF or the approximate variants (see Section 6.2 and 6.2) may be used for this initialization. Note from Algorithm 6 that the error criterion $\mathcal{E}(\cdot)$ in this second stage which inserts the rest of the $m - \tilde{m}$ rows is performing an exhaustive search. One can get around the exhaustive search using ideas from Section 6.2, however, the gain in speed or performance in minimal. Overall, the incremental procedure scales efficiently for very large matrices as we will see shortly in the experiments, compared to using the batch-wise scheme on the entire matrix.

Algorithm 8 INCREMENTAL MMF($\mathbf{C}$)

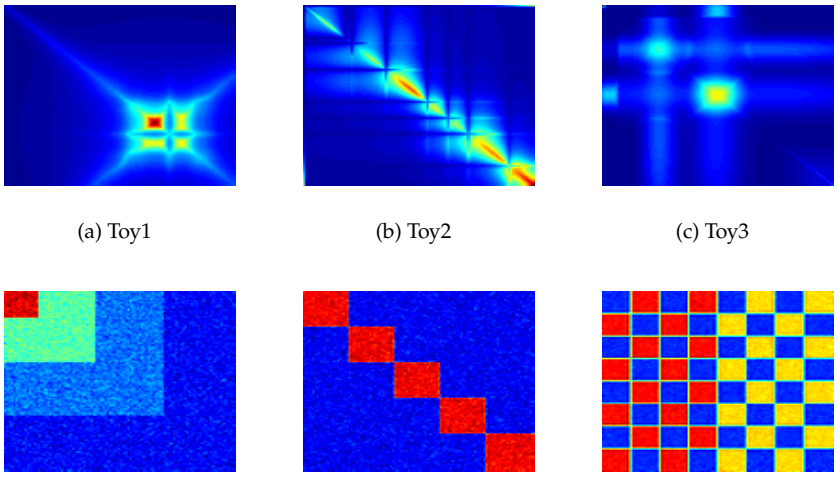
Output: $\mathcal{M}(\mathbf{C})$
$\bar{\mathbf{C}} = \mathbf{C}_{[\tilde{m}],[\tilde{m}]}$, $L = m - k + 1$
$\{t^\ell, s^\ell\}_1^{\tilde{m}-k+1} \leftarrow$ BATCHMMF($\bar{\mathbf{C}}$)
for $j \in \{\tilde{m}+1, \ldots, m\}$ **do**
$\{t^\ell, s^\ell\}_1^{j-k+1} \leftarrow$ INSERTROW($\bar{\mathbf{C}}, \mathbf{C}_{j,:}, \{t^\ell, s^\ell\}_1^{j-k}$)
$\bar{\mathbf{C}} = \mathbf{C}_{[j],[j]}$
end for
$\mathcal{M}(\mathbf{C}) := \{t^\ell, s^\ell\}_1^L$

6.4 Evaluations

Recall the original motivation for constructing multi-scale factorization of symmetric matrices – designing a general purpose tool for modeling hierarchical structure in the data (e.g., for use in selecting, ranking or exploration of hypotheses; see Section 6.1). Hence, we perform exhaustive evaluations primarily in the context of clinical trials design, but also for other application domains including computer vision. These evaluations are three fold. The core component that incremental MMF relies on is that an existing MMF can be 'modified' appropriately by adjusting the k-tuple choices. Clearly, the first set of evaluations then need to justify this incremental procedure for computing MMF on any given matrix. Hence, using a variety of simulated symmetric matrices (each with distinct hierarchical structures – see Figure 6.1 from Section 6.1), we compare the error incurred by incremental MMF compared to batch version. This next two evaluations setups focus on – MMF Scores and MMF Graphs. MMF scores are generalized statistical leverage scores that assign importance (ranking) to the rows/columns of the given matrix, while MMF graphs (as introduced already in Section 6.3) are useful visualizations of the factorization. The experiments on MMF graphs in particular will utilize computer vision datasets because they are easy to interpret (e.g., a cat is more *related* to dog in comparison to a horse) and thereby easier to infer the higher-order relationships in the factorization.

Incremental MMF versus Batch MMF

The first set of evaluations compares the incremental MMF to the batch version. In other words, given some $\mathbf{C}$, we are comparing the error incurred (see (6.3)) by the batch MMF from Algorithm 5 to the incremental MMF from Algorithm 8. The two components of the algorithm relevant for these evaluations are the initialization step and the insertion criteria. The presentation in Algorithms 8 and 7 uses the batch MMF and exhaustive search for insertion. This would be the ideal version of the incremental MMF – because one can replace either of these steps with two approximate alternatives described in Section 6.2. This leads to as total of 9 different incremental MMF algorithms, and we will use six different toy matrices to test these versions against batch MMF. The data matrices are shown in Figure 6.4, each of which is distinct in terms of the hierarchical (and blocky or otherwise) structure, and these 6 examples constitute many of the real-world datasets that we use later in Sections 6.5 and 6.6. Recall that MMF



(a) Toy1 (b) Toy2 (c) Toy3

(d) Toy4 (e) Toy5 (f) Toy6

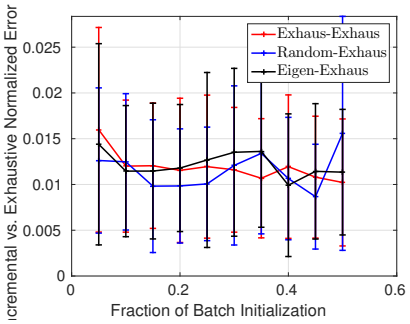
Figure 6.4: Symmetric matrices from six different toy datasets – Toy1 to Toy6.

error is the off-diagonal norm of $\mathbf{R}$, except for the $\mathcal{S}_L \times \mathcal{S}_L$ block (see (6.1)), and the smaller the error is, the closer the factorization is to being exact (see 6.1). Constructing MMFs on Figure 6.4 showed that the incremental MMFs incur approximately the same error as the batch versions, while achieving $\gtrsim 20$ times speed-up compared to a single-core implementation of the batch MMF (see Figure 6.6). Figures 6.5 and 6.7 show the results on Toy1 matrix in Figure 6.4, since the trends are similar for all the 6 matrices considered (including 3 ADNI and WRAP covariance matrices constructed from real data later). Specifically, the loss in factorization error is $\lesssim 4\%$ of $\|\mathbf{C}\|_{\text{Frob}}$, with no strong dependence on the fraction of the initialization $\tilde{m}$ (see Algorithm 8).

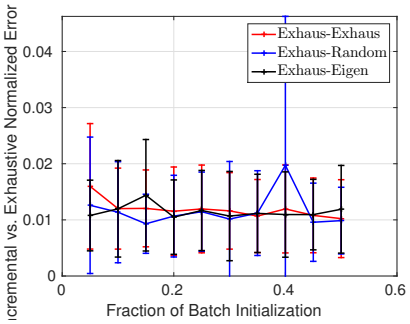
6.5 MMF Scores

The objective of MMF (from (6.2)) is the signal that is not accounted for by the k^{th}-order rotations of MMF, and so it is 0 whenever $\mathbf{C}$ is *exactly* factorizable. Hence, $\|(\mathbf{C} - \mathcal{M}(\mathbf{C}))_{i,:}\|$ is a measure of the extra information in the i^{th} row that cannot be reproduced by hierarchical compositions of the rest. Such value-of-information summaries, referred to as *MMF scores*, of all the dimensions of $\mathbf{C}$ give an importance sampling distribution, similar to statistical leverage scores

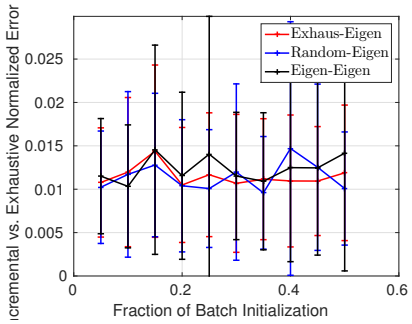
(Boutsidis et al., 2009; Ma et al., 2015). These samplers drive several regression tasks in vision including gesture tracking (Rautaray and Agrawal, 2015), face alignment/tracking (Cao et al., 2014) and medical imaging (Friston et al., 1994). Moreover, the authors in (Ma et al., 2015) have shown that statistical leverage type marginal importance samplers may not be optimal for regression. On the other hand, MMF scores give the conditional importance or *relative leverage* of each dimension/feature given the remaining ones. This is because MMF encodes the hierarchical block structure in the covariance, and so, the MMF scores provide better importance samplers than statistical leverages. We demonstrate the utility of these scores on ADNI and WRAP datasets – see the discussion in Section 2.3 and 2.3. Using 1300 instances accumulated between the two datasets, Figure 6.8(a,b) shows the instance covariance matrices after selecting the 'best' 5% ROIs among 80 such regions (see Figure 6.8(c)) using the classical leverage



(a) Error vs. Enhaus Insert

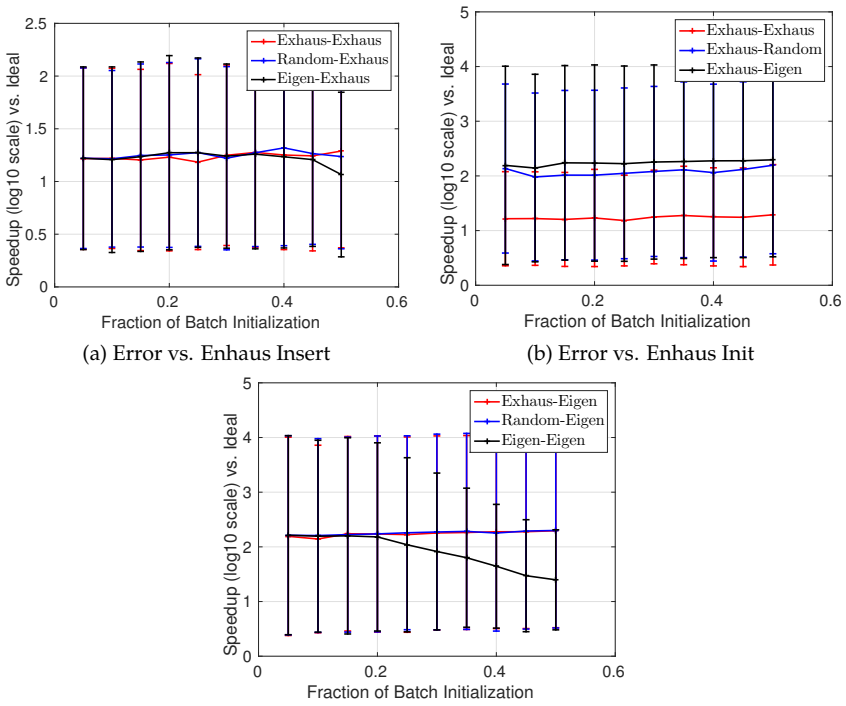


(b) Error vs. Enhaus Init



(c) Error vs. Eigen Insert

Figure 6.5: Incremental versus Batch MMFs (Toy1): Error Trends



(a) Error vs. Enhaus Insert

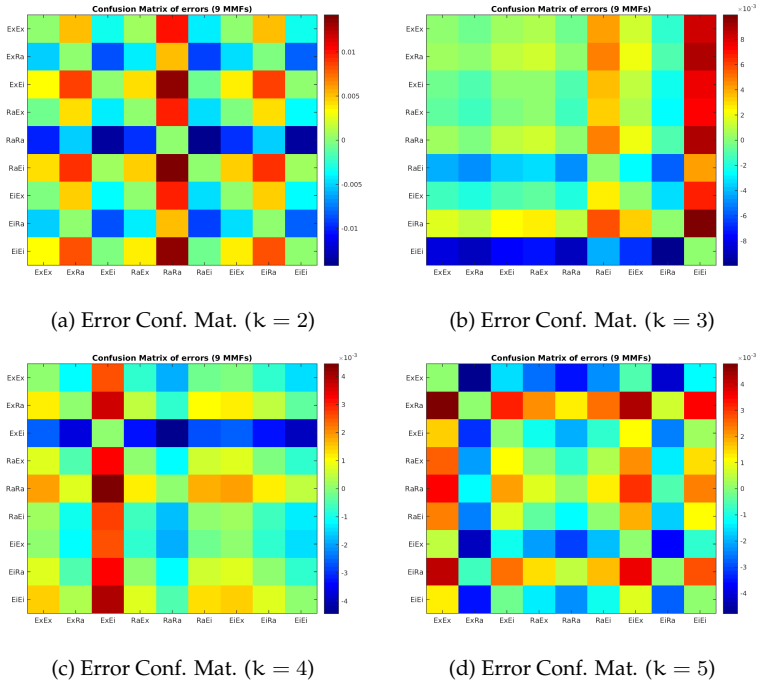
(b) Error vs. Enhaus Init

(c) Error vs. Eigen Insert

Figure 6.6: Incremental versus Batch MMFs (Toy1): Speed-up Trends

scores and MMF scores respectively. The block structure representing the two classes, diseased and non-diseased, is clearly more apparent with MMF score sampling (see the yellow block versus the rest in Figure 6.8(b)).

Despite the fact that the features have high degree of block structure (covariance matrix from Figure 6.8(c)), this information is rarely, if ever, utilized within the downstream models, say multi-linear regression for predicting health status. To further evaluate the notion of using MMF scores for *selection* – a version of enrichment if the application is clinical trials – we train a linear model with the goal of using a *fraction* of the voxel ROIs coming from PET imaging data sampled according to statistical leverages (from (Boutsidis et al., 2009)) and relative leverage from MMF scores. Unlike LASSO, the feature samplers are agnostic to the responses. This is a setting similar to optimal experimental design (de Aguiar et al., 1995) – see the extended discussion about enrichers in Chapter 4, specifically Sections 4.3 and 4.4. This general response-agnostic setting makes the presented results relevant for other tasks like ranking feature



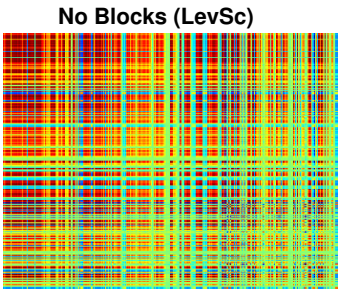
(a) Error Conf. Mat. ($k = 2$) (b) Error Conf. Mat. ($k = 3$)

(c) Error Conf. Mat. ($k = 4$) (d) Error Conf. Mat. ($k = 5$)

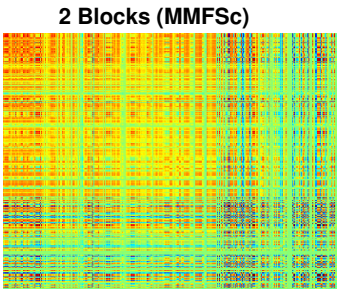
Figure 6.7: Incremental versus Batch MMFs (Toy1): Confusion matrices

importance for hypothesis design, outlier detection (and data pre-processing) etc. The inputs are ROIs from imaging data and the responses are AD biomarkers including *disease status*, *age* and *genotype*. A bigger set of evaluations were performed, the results being presented here are representative of all the trends.

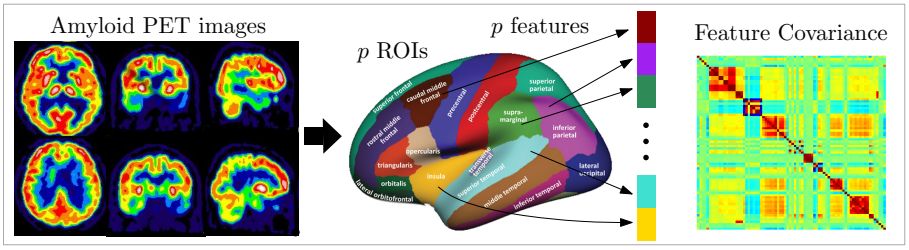
Figures 6.9 and 6.10 presents a summary snapshot of all evaluations performed on the MMF scores. The regression model here uses WRAP ROIs as predictors and the corresponding disease status as response. The size of initialization in the incremental MMF for all the evaluations is $\tilde{\mathfrak{m}} = 0.1\mathfrak{m}$. As shown by the blue versus black curves in *all* of these plots, the voxel ROIs picked by MMF are better both in terms of adjusted-R^2 (the explainable variance of the data) and F-statistic (the overall significance). More importantly, the first few ROIs picked up by MMF scores are more informative than those from leverage scores, and this trend has been consistent across both ADNI and WRAP data and all three responses. Nevertheless, the F-statistic and R^2 saturate at different fractions (on the x-axis) for different responses, implying that the



(a) LevScore Sampling



(b) MMFScore Sampling



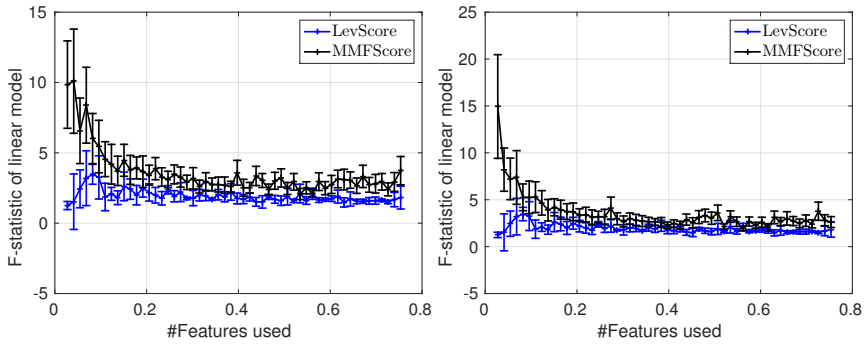
(c) Regression Setup – Constructing the ROI data

Figure 6.8: Feature Importance Sampling of MMF Scores vs. Leverage Scores

benefit of sampling using MMF scores, albeit present, varies depending on the downstream application. The evaluations presented here included sampling features based on their relative leveraging. Similar sampling can be performed for instances thereby resulting in enrichment designs in clinical trials using input data covariance. Further, we restricted ourselves to evaluating regression models after such relative leverage sampling, instead one can evaluate classification performance or any other scenario. We note that ROI features and regression were used because they are easier to interpret, and also to present the *general purpose* usage of MMF scores beyond their applicability for subject selection in clinical trials.

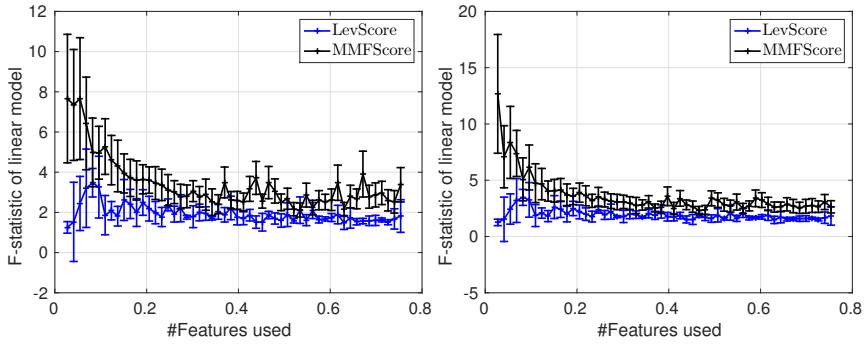
6.6 MMF Graphs

The concept of MMF graphs was already introduced earlier in Section 6.3 while presenting the algorithm for incremental MMF. Refer to Figure 6.3 where an example MMF graph is shown for a 3^{rd}-order MMF computed on a 5×5



(a) F-stat Comp; k = 8

(b) F-stat Comp; k = 12



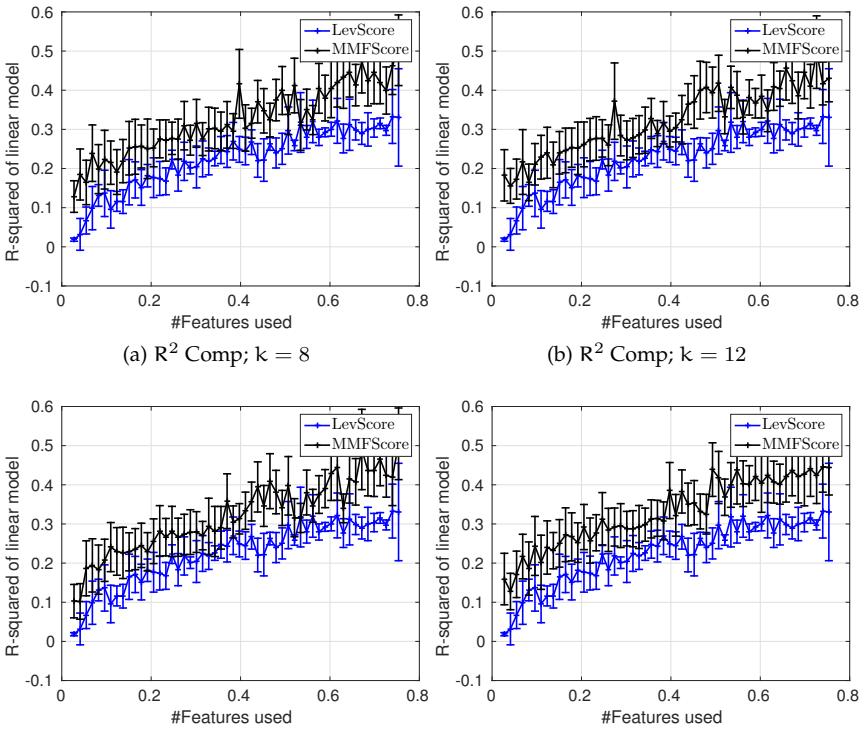
(c) F-stat Comp; k = 16

(d) F-stat Comp; k = 20

Figure 6.9: Importance Sampling: MMF Scores vs. Leverage Scores: F

symmetric matrix. This is a powerful exploratory tool, and we demonstrate that on a variety of datasets here. Note that the broad goal here is to compute MMFs, visualize them via the MMF graphs and interpret them. Hence, to ensure that the interpretation is 'easier', we first show exhaustive results on computer vision datasets, followed by MMF graphs on cognitive and/or imaging features from ADNI and WRAP datasets.

Computer Vision Dataset: The vision dataset used here is ImageNet (Russakovsky et al., 2015). This is a large database of over one thousand different color images of objects/categories primarily acquired for benchmarking object detection algorithms. Clearly, visualizing MMF graphs on dozens of classes is non-trivial and useless to interpret, and hence, the presentation will restrict to class-by-class covariance matrices of ten or so classes from the 55 classes and 25 attributes listed in Table 6.1.



(a) R^2 Comp; $k = 8$ (b) R^2 Comp; $k = 12$

(c) R^2 Comp; $k = 16$ (d) R^2 Comp; $k = 20$

Figure 6.10: Importance Sampling: MMF Scores vs. Leverage Scores: R^2

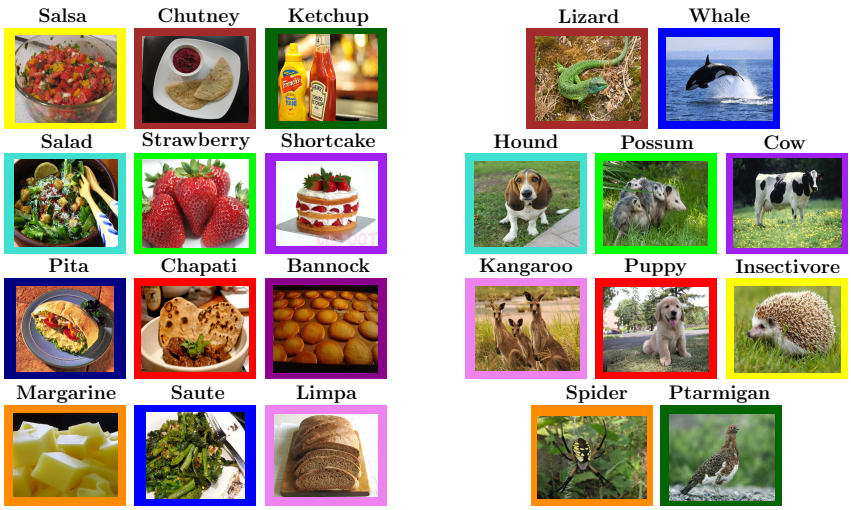
55 **classes:** bag, ball, basket, basketball, bathtub, bench, blackboard, bottle, box, building, chalkboard, edifice, saute, margarine, bannock, chapati, pita, limpa, shortcake, strawberry, salad, ketchup, chutney, salsa, puppy, green lizard, garden spider, ptarmigan, kangaroo, possum, cow, insectivore, killer whale, hound, male horse, warhorse, pony, mule, zebra, bison, sheep, goat, blackbuck, deer, elk, pot, rack, roller coaster, rule, sail, sheet, ski, couch, racket, stick, table, toilet seat 25 **attributes:** black, blue, brown, furry, gray, green, long, metallic, orange, pink, rectangular, red, rough, round, shiny, smooth, spotted, square, striped, vegetation, violet, wet, white, wooden, yellow

Table 6.1: 55 classes and 25 attributes from ImageNet.

The evaluation setup involves computing MMFs on class covariance matrices. Let m be the number of classes/attributes we are interested in, and $\mathbf{C} \in \mathbb{R}^{m \times m}$ be the class covariance matrix. For each class i, let $p_1, \ldots, p_{|i|}$ denote the indices of the data instances belonging to class i (for large classes at most a randomly chosen $2k$ instances are used for brevity). Each entry of $\mathbf{C}$ is

$$\mathbf{C}_{i,j} = \frac{1}{|i||j|} \sum_{\hat{i}=p_1}^{p_i} \sum_{\hat{j}=q_1}^{q_j} \mathbf{d}_{\hat{i}}^T \mathbf{d}_{\hat{j}} \tag{6.8}$$

where $\mathbf{d}_{\hat{i}}$ is the representation of $\hat{i}^{th}$ data instance coming from some hidden layer of the given deep network. These representations can correspond to hand-designed features like SIFT (Lowe, 1999) or hidden/output representations from state-of-the-art deep networks including VGG network (Chatfield et al., 2014), GoogLeNet (Szegedy et al., 2014), ResNet (He et al., 2016), DenseNet (Huang et al., 2016) etc. MMF graphs would then be constructed on $\mathbf{C}$ as described in Sections 6.3 and 6.3, and for the most past we will look at the two sets of classes shown in Figure 6.11.



(a) 12 **food classes** (b) 10 **animal classes**

Figure 6.11: The food and animal classes for studying MMF graphs

Decoding deep representations

The first set of results considers representations from GoogLeNet on the food classes from Figure 6.11. We will extensively study the resulting MMF graphs from $\mathbf{C}$s computed (via (6.8)) for different layers of this deep network and for different orders (k). Figure 6.12(a) visualizes a 5^{th} order MMF graph learned on class covariance matrix from the FC7 layer which is the last hidden layer that feeds into softmax (FC: Fully connected layer).

The semantics of breads and sides. The 5^{th} order MMF says that the five categories – *pita*, *limpa*, *chapati*, *chutney* and *bannock* – are most representative of the localized structure in the covariance. Observe that these are four different flour-based main courses, and a side *chutney* that shared strongest context with the images of *chapati* in the training data (similar to the body building and dumbell images from (Mordvintsev et al., 2015)). MMF then picks *salad*, *salsa* and *saute* representations' at the 2^{nd} level, claiming that they relate the strongest to the *composition* of breads and *chutney* from the previous level (see visualization in Figure 6.12(a)). Observe that these are in fact the sides offered/served with bread. Although VGG-S was not trained to predict these relations, according to MMF, the representations are inherently learning them anyway – a fascinating aspect of deep networks i.e., they are seeing what humans may infer about these classes.

Any dressing? What are my dessert options? Let us move to the 3^{rd} level in Figure 6.12(a). *margarine* is a cheese based dressing. *shortcake* is dessert-type meal made from *strawberry* (which shows up at 4^{th} level) and bread (the composition from previous levels). That is the full course. The last level corresponds to *ketchup*, which is an outlier, distinct from the rest of the 10 classes – a typical order of dishes involving the chosen breads and sides does not include hot sauce or ketchup. Although *shortcake* is made up of strawberries, "conditioned" on the 1^{st} and 2^{nd} level dependencies, it is less useful in summarizing the covariance structure. An interesting summary of this hierarchy from Figure 6.12(b) is – an order of *pita* with side *ketchup* or *strawberries* is atypical in the data seen by these networks.

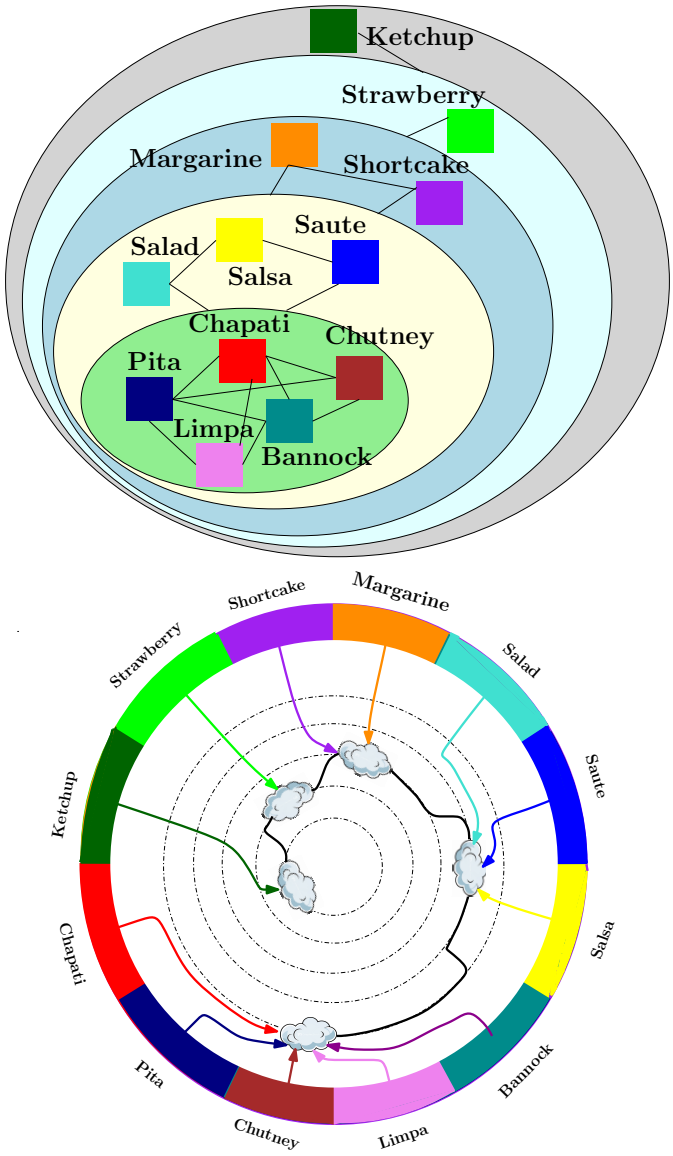


Figure 6.12: Class hierarchy using GoogLeNet DepthConcat9 reps with 5^{th} order MMF

Are we reading tea leaves?

There are many questions that arise right away based on the inference drawn about the MMF graph in Figure 6.12. The second set of evaluations that we

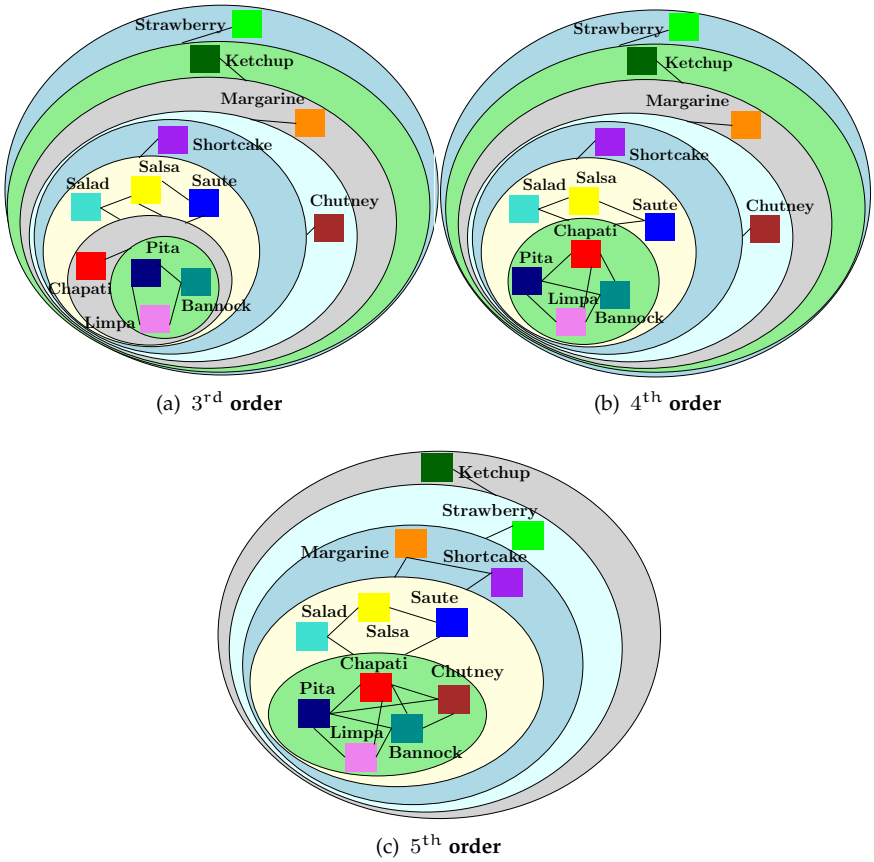
discuss concern these questions.

Hierarchy is agnostic to task: First note that GoogLeNet is not trained to learn the hierarchy of categories, or the loss function was not taking into account any hierarchical information what-so-ever. This is unlike the alternate works from (Yan et al., 2015; Dong et al., 2017; Badrinarayanan et al., 2015), and the task that learned GoogLeNet weights correspond to is simply object/class detection. Hence, the relationships MMF graph inferred are completely a by-product of the power of deep networks to learn contextual information, *and* the ability of MMF to model these compositions by uncovering the structure in the covariance matrix. Nevertheless it is reasonable to ask if this description is meaningful since the semantics drawn above are subjective, and so we continue with evaluating more aspects.

Choice of the MMF order: The first aspect that one may ask is if the compositions are sensitive/stable to the order $\mathtt{k}$ – a critical hyperparameter of MMF. Figure 6.13 shows that graphs from $\mathtt{k} = 3, 4$ and 5 respectively, and the resulting hierarchies are similar to that from Figure 6.12. Specifically, the different breads and sides show up early, and the most distinct categories (*strawberry* and *ketchup*) appear at the higher levels.

Choice of the deep network: The next question to ask is whether the MMF graph structure varies "too much" when the representations are picked from different deep networks. Figure 6.14 shows the graphs from VGG-S network (first) and Residual network (third) in comparison to the GoogLeNet MMF graph. The graphs are computed with $\mathtt{k} = 5$. It is reasonable to expect some difference because the networks themselves are very different to each other in terms of architecture, i.e., the depth, and convolutional versus inception versus residual – however, the overall trend of relationships are still similar to those discussed in Section 6.6

Baseline comparison: We compared MMF's class-compositions to the hierarchical clusters obtained from agglomerative clustering of representations. The categories like *bannok* and *chapati*, or *saute* and *salad* are closer to each other here. However, the compositional relationships (like the dependency of *chutney/salsa/salad* on several breads, or the disparity of *ketchup* from others) shown in Figures 6.12(a) and 6.13, are not apparent in these dendrogram outputs.



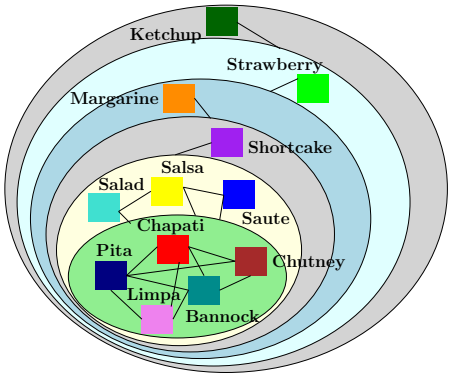
(a) 3^{rd} **order** (b) 4^{th} **order**

(c) 5^{th} **order**

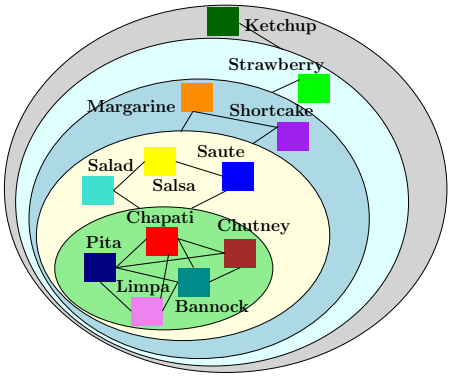
Figure 6.13: Class hierarchy using GoogLeNet with different MMF orders

The generality of MMF graphs

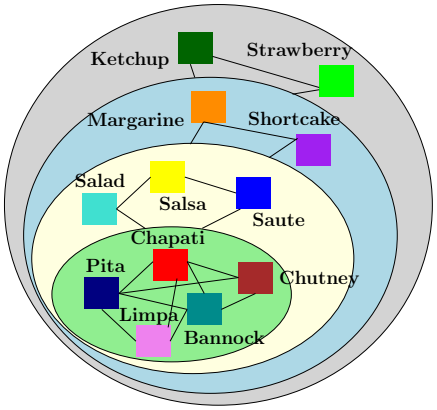
Building upon the investigation of MMF graphs presented in the earlier section, we may evaluate the generality and *applicability* of MMF graphs by checking whether the hierarchical relationships coming from MMF graphs are dependent on the choice of the classes being studied, and if so, will humans be able to discern that; or more interestingly whether they can directly tell us something non-trivial about the deep network being used. Note that all these aspects are to be studied thoroughly before using MMF graphs for *difficult to interpret tasks* like subject selection in clinical trial enrichment, or hypotheses (and outcome) design for measuring trial efficiency.



(a) **VGG-S (Last hidden layer.)**



(b) **GoogLeNet (Last hidden layer.)**



(c) **ResNet (Last hidden layer.)**

Figure 6.14: Class hierarchy using VGG-S, GoogLeNet and ResNet

Choice of the classes being studied:

Figure 6.16 shows MMF graphs on other families of classes from Imagenet (see Section 6.6) Figure 6.16(a) contains 10 different versatile set of classes i.e., the contextual information (like background, typical sets of objects that are found etc.) for *cow* or *hound* may be drastically different from *kangaroo* or *ptarmigan*. These classes are from Figure 6.11(b). Clearly, the most distinct classes – *green lizard*, *killer whale* are resolved at highest levels. The composition at first level involves those classes which share context – there is most certainly grass or trees on the bark in these images. Given the first level, Figure 6.16(a) shows that *kangaroo* rather than *green lizard* is most related to this first level

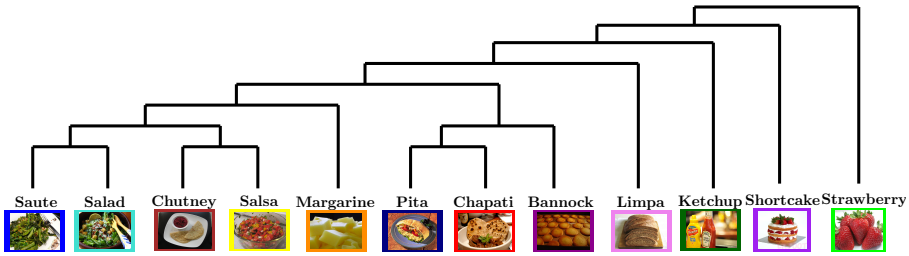
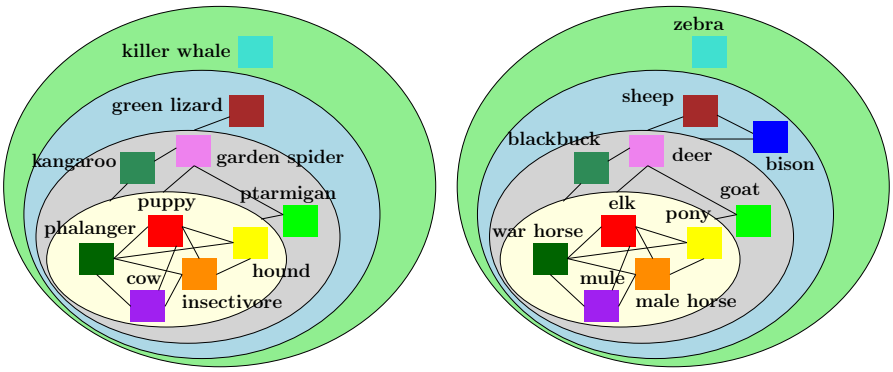


Figure 6.15: Agglomerative clustering on GoogLeNet DepthConcat9 representations

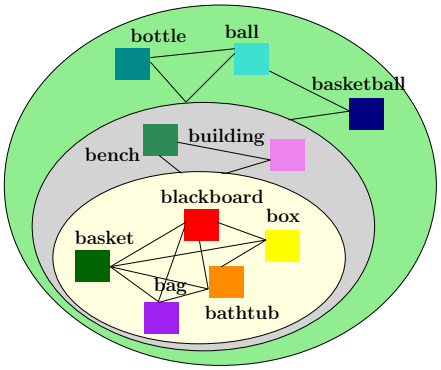
composition – implying a clear sense of hierarchy in learned representations. The categories in Figure 6.16(b) that MMF picked for first level compositions are highly correlated to begin with, both with respect to the texture/pixel-values as well as the background (most show up with grassy or landscape background). It is reasonable to say that human perception would imply that *blackbuck*, *deer*, *goat* are most related to the first level compositions – which were composed at the next level. *zebra* clearly is the most distinct one, implying that many compositions of the first and second level categories may not really infer what constitutes a *zebra*.

In another example, 8 of the 12 classes from Figure 6.16(c) are sports accessories/objects and the rest (*pot*, *couch*, *sheet*, *toilet seat*) are household-type categories. This was done to see what the MMF will do, if it is forced to pick a hierarchy among arbitrary classes. Clearly, *pot* shows up at the last level. The first level compositions are 5 of the sports related classes. Interestingly *couch* is closer in composition to the first level than a *roller coaster* or *sail*, which may be because the context of four of the first level classes is that they are 'indoors'. *sail* and *roller coaster* have sky and water as background respectively, making them distinct enough from the first level composition, and pushing *couch* or *toilet seat* instead. Further, if the networks are as powerful as we hope them to be – and the MMF is in fact picking up interesting hierarchical relationships, then this class-wise trained network should capture interesting, human-releatable, compositions for meta-class attributes like *color* or *shape*. Figure 6.16(d,e) show the MMF graphs from FC7 representations for these *color* and *shape* attributes. This most distinctive of the colors (like *red*, *yellow*) are picked up at lowest level. Bright colors like *pink* and *violet* which are in fact compositions of basis

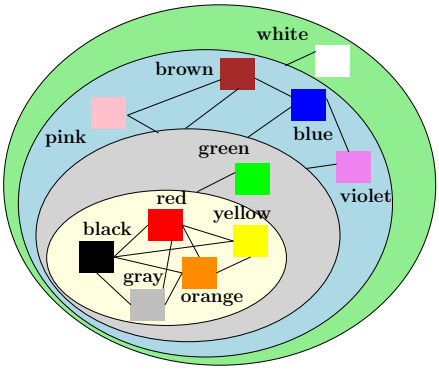


(a) **Animal Classes**

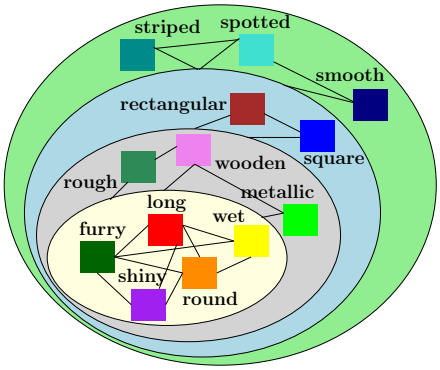
(b) **Correlated Classes**



(c) **Another Example**



(d) **Color attributes**

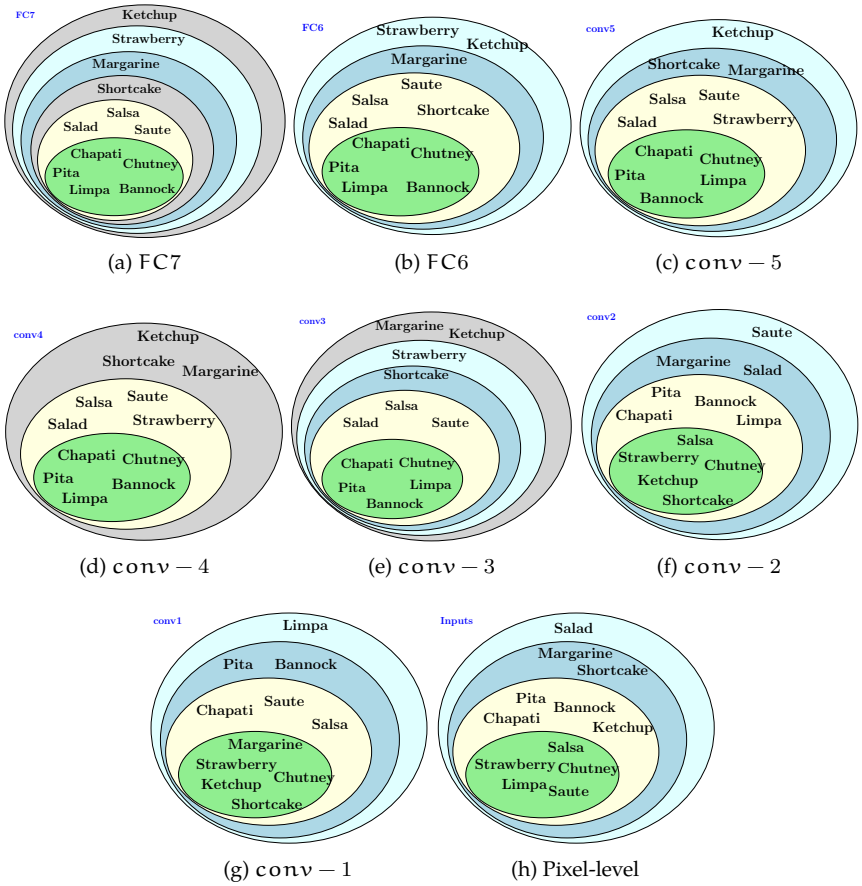


(e) **Shape attributes**

Figure 6.16: Class hierarchies on variety of ImageNet classes/attributes

RGB colors are at higher levels of the MMF graph. Similarly, visually the most

evident shapes like *furry* and *shiny* form lower level composition, while most subtle shapes like *smooth* and *spotted* are at the last level.



(a) FC7 (b) FC6 (c) conv − 5

(d) conv − 4 (e) conv − 3 (f) conv − 2

(g) conv − 1 (h) Pixel-level

Figure 6.17: Class hierarchies from different VGG-S layers

Changing hidden layer – The flow of MMF graphs

For the last set of results to evaluate MMF graphs, we will look at the relationship between multiple graphs and how different classes relate across them. If the class relationships from Figures 6.12, 6.13 and 6.14 are are non-spurious, then similar trends should be implied by MMF's on different (higher) layers of the networks. Figures 6.17 and 6.18 shows the MMF graphs of all the hidden layers on food and animal classes from Figure 6.11 with representations from VGG-S and GoogLeNet respectively. Each of the two sets of graphs are in

Figure 6.18: Class hierarchies from different GoogLeNet layers

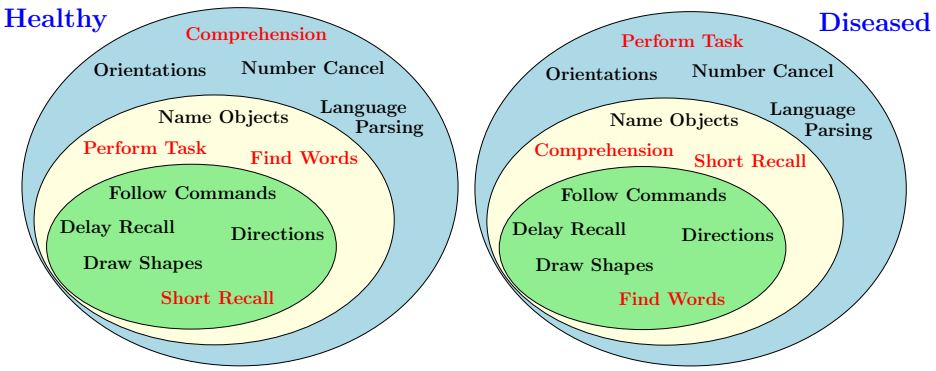
descending order starting from the last layer that feeds softmax all the way to inputs. Pixel-level features are non-informative (the success of deep learning is a direct evidence for this), and clearly, the classes whose RGB values/signals correlate are at $\ell = 0$ in both Figure 6.17(h) and Figure 6.18(l). With regard to the graphs for the other layers, there are several interesting things happening here. First observe from Figure 6.17 that the strongest compositions, the 8 classes from $\ell = 1$ and 2, are already picked up half-way thorough the VGG-S, providing further evidence that the compositional structure implied by MMF is data-driven. Most importantly, the first four graphs from Figure 6.17, which are from the FC7, 6, 5^{th} and 3^{rd} convolutional layer respectively, we see that $\ell = 1$ and 2 have the same compositions. Secondly, this says that the strongest

possible representative classes may be learned in lower layers of the network, and if these 8 classes are a priority, then the predictions of the VGG-S' 3^{rd} convolutional layer may already be *good enough*.

Third there are some classes that "move around" multiple layers of the MMF as the hidden layer changes from inputs to outputs. Such classes are indicated in 'blue' in Figure Figure 6.18 i.e., classes shown in blue at a given layer moved their MMF level contingency compared to previous layer. For instance, the cow class from Figure 6.18(l). These classes represent *band-pass frequencies* in terms of correlational spectrum of the given classes, and hence serve as median representative class of the data. Lastly, if there is large variance among the compositions beyond simply one class like in the previous example, especially in lower MMF graph levels, then the lower layers of the deep network may not have efficiently accounted for the strongest representative classes. This implies that the hidden layer lengths or activations for lower network layers need to be modified. A smaller variance however may imply that adding additional layer may be beneficial. Visualizations like Figures 6.17 and 6.18 are a powerful exploratory tool in decoding, and possibly, designing deep networks (Ithapu et al., 2017b) – providing solutions to our original motivation of interpretability of deep representations. Apart from visualizing deep representations, these evaluations show that MMF graphs are vital exploratory tools for category/scene understanding from unlabeled representations in transfer and multi-domain learning (Ben-David et al., 2010). This is because, by comparing the MMF graph *prior* to inserting the new unlabeled instance to the one *after* insertion, one can infer whether the new instance contains non-trivial information that cannot be expressed as a composition of existing categories.

MMF graphs for Clinical Trials Design

The evaluations and results in Sections 6.6, 6.6 and 6.6 show that MMF graph is a very useful, general purpose tool, for modeling hierarchical structure in the data, for inferring/summarizing inter-class relationships, and for modeling *change* in the relational structure across sets of covariances. Since all the conclusions drawn about the MMF graph topology have been validated with the vision datasets and interpretable class-wise relationships, we will now simply apply MMF graphs to ADNI and WRAP datasets and show their applicability to the aspect of discovering new combinations of features for outcome/enricher design in clinical trials. Note that selection and ranking of features or instances

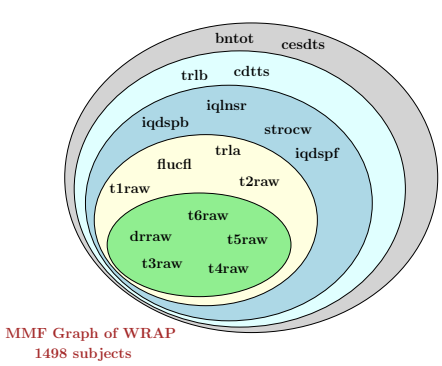


(a) MMF graph on healthy (b) MMF graph on diseased

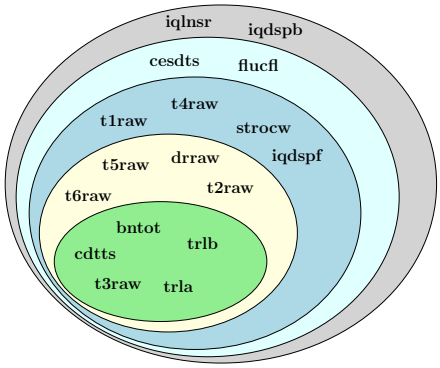
Figure 6.19: MMF graphs constructed on cognitive questionnaire

can also be done using MMF graphs – the contingency level of a given entity represents its importance (recall that MMF graph levels are directional). Nevertheless, since we already discussed MMF scores for such constructs, we will focus on the above aspect instead. Figure 6.19(a,b) shows the MMF graphs computed on healthy and diseased subjects. Each of the two graphs is constructed on feature-by-feature covariance matrix computed from 12 items on the ADAS cognitive questionnaire (see Section 2.3 for the details). The items that moved within the graph between healthy and diseased subjects are marked in red color.

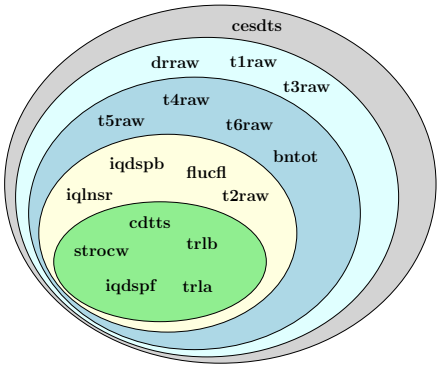
The MMF graphs computed on 12 different cognitive scores (not items of a questionnaire) acquired as a part of WRAP study are shown in Figure 6.20(a) – please refer to Section 2.3 for definitions of these scores. This is computed form score-by-score covariance on 1490 subjects who were part of the WRAP study. Figures 6.20(b–e) correspond to the "difference" graphs. The WRAP subjects were first binned according to their age < 45, $46 - 55$, $56 - 65$, $66 - 75$ and > 75 respectively, and the difference of covariance matrices with respect to < 45 subjects were constructed i.e., the difference of two score covariance for $46 - 55$ and < 45, and so on. The MMF graphs on the resulting four difference of covariances are shown in Figures 6.20(b–e). These people are *presymptotic*, or, middle-aged and older adults who have not yet developed complete signature of AD. Hence, the subtle changes in the ordering of the scores, and the MMF levels to which they correspond to, is a computational signature of changes in decline. If one intends to design an outcome for conducting a trial on these



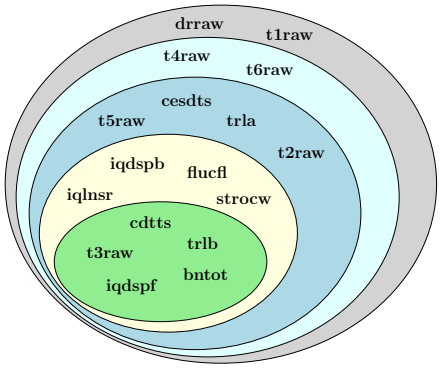
(a) MMF graph on WRAP Scores



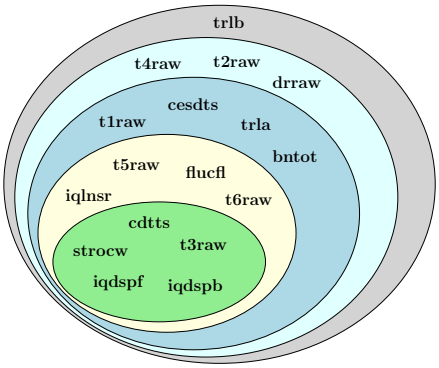
(b) $46 - 55$ vs. < 45



(c) $56 - 65$ vs. < 45



(d) $66 - 575$ vs. < 45



(e) > 75 vs. < 45

Figure 6.20: MMF graphs constructed on WRAP cognitive scores

subjects, one can simply use the scores that have high variance across the five graphs. Similarly, if we intend to filter out the adults who do not show decline, one can first compute subject-wise MMF graphs estimated from each subjects's individual scores (rank-1 covariance matrices). We then look for subjects whose graph is similar to the 'average' MMF graph from Figure 6.20(a), and remove them from the trial population. An alternative would be to compute the MMF graphs on difference of covariance matrices, where the two matrices correspond to different time points, i.e., the subject's visits to the clinic (similar to the graphs from Figures 6.20(b-e)). The individuals whose graphs are closest to the healthiest MMF graph from Figure 6.20(b) will then be filtered out. A variety of other, more quantifiable, approaches can be utilized as well to design such outcomes and enrichers, and we will discuss about these briefly later in Section 7.1.

Chapter 7

Conclusions

In this concluding chapter, we briefly revisit the context and contributions of the two parts of the thesis, and provide pointers to some of the future directions and open questions for the algorithms from Chapters 3–6. Recall the setup of this report from Chapter 1 and Figure 1.3. We are broadly interested in computational algorithms for designing clinical trial outcomes and enrichment criteria for study population. In chapter 3, a fast and robust multiple hypothesis testing procedure was presented with applications in testing hypothesized outcomes; in chapter 4, an optimal enrichment criterion is proposed using ensemble of deep networks; in chapter 5, a new analysis framework and sets of technical results are proposed for designing site-specific deep networks in multi-site studies and clinical trials; and lastly in chapter 6, a multiscale matrix factorization procedure is presented with applications in instance/feature selection and exploratory hypothesis design.

7.1 Future Directions

The fast multiple hypothesis testing algorithm proposed in Chapter 3 – rapid permutation testing – exploits redundancy (correlations) across multiple test statistics computed for different labeling of the groups, and the modeling assumption is that the statistics can be expressed as a summation of a low-rank basis component and a high-frequency residual. The evaluations showed that the proposed testing procedure preserves sensitivity. Building upon Chapter 3, there are a few interesting directions for further validating the proposed testing algorithm.

1. Prior information about the correlational structure, beyond the empirical observation that the testing matrix is low-rank plus sparse, may improve

the estimation procedure specifically in the training module.

2. Two sample tests have been the main focus of our presentation, and a natural question to ask would be whether RapidPT can extend to far complex permutation testing designs (including linear models) – and possibly to non-permutation based boostraping designs.

The focus of a large fraction of clinical trials is testing the drug's effect i.e., the treatment effect on *two* groups – subjects who are given the treatment versus the placebo group. Hence, the discussion of optimal enrichment criterion in chapter 4 is motivated using a two-sample t test and the corresponding sample size calculations. Such optimal criteria can be designed for other types of trial designs, for instance, using multi-treatment trials where > 1 treatment groups (with different treatment), using alternate statistics like χ^2, F etc. Also, ensemble methods beyond simple averaging schemes, like adaptive boosting, are known to be robust and further increase sensitivity. Extending randomized deep networks using such adaptive frameworks may provide solutions to small sample deep learning, leading to new avenues of deploying neural network learning models in biology and medicine with very small datasets (less than a dozen or so instances).

The contributions of Chapter 5 are motivated from designing deep networks, specifically network architectures that learn the data 'well', while keeping into account the restrictions on typical medical and clinical datasets (like multi-site acquisition, deployability by non-experts etc.). The treatment of the presentation was technical since the first set of steps in addressing this design choice problem were to provide algorithms that perform architectural model selection given only the dataset and some meta constraints. Clearly, the next logical directions include comparing and *combining* the proposed design choice procedures to existing hyperparameter selection techniques like Hyperband Li et al. (2016b) and Spearmint Feurer et al. (2015). In particular, the proposed framework targets convergence of the architectures without direct consideration of the generalization performance. Hence, it is reasonable to construct *a prior* on architectural choices using the bounds from Chapter 5, and evaluate resource allocation procedures like Hyperband on these prior set of choices. Among other open questions include extending the framework to convolutional architectures as well as improving (tightening) the bounds.

One of the main goals of the proposed incremental multiresolution matrix factorization (MMF) algorithm (from Chapter 6) is to construct wavelets on symmetric matrices without the computational overhead (which involved com-

binatorial search prior to the proposal). In Sections 6.5 and 6.6 we showed a variety of applications for this factorization, and building upon these approaches, the following directions are of interest for incremental MMF extensions.

1. Evaluating the hierarchical compositions inferred by MMF are critical for using it beyond simple selection or ranking applications. A large-scale study (for instance using Amazon Mechanical Turk) will provide evidence for both generalizing the model and its relation to human's perception of compositional structure.
2. Defining metrics (distances) between two MMF graphs is the natural extension for solving a variety of problems in hierarchical modeling, for instance, equivalence classes of hierarchical summaries, analyzing stream (or flow) of covariance structures (like evolving graphs, social webs, disease progression etc.). Since MMF graph corresponds to sequence of sparse rotations, such metrics and equivalence classes can be defined with sets of instances or features, i.e., MMF distances between trial subjects and the disease markers. Further, equivalence classes of object/category hierarchies for learning problems from computer vision is of independent interest.

In conclusion, the contributions of this thesis show evidence for applying machine learning algorithms beyond computational diagnosis. The proposed algorithms show that machine learning models can be customized for designing clinical trials and targeting interventions. There is further optimism at this interface towards targeted drug design, customized treatment and precision medicine, within the landscape of Alzheimer's Disease as well as other biological disorders.

List of References

A. Eklund, D. Forsberg S.M. LaConte, P. Dufort. 2013. Medical image processing on the gpu - past, present and future. *Medical Image Analysis* 17:1073–1094.

A. Eklund, H. Knutsson, M. Andersson. 2012. fmri analysis on the gpu-possibilities and challenges. *Computer Methods and Programs in Biomedicine* 105:145–161.

Abdel-Hamid, Ossama, Li Deng, and Dong Yu. 2013. Exploring convolutional neural network structures and optimization techniques for speech recognition. In *Interspeech*, 3366–3370.

de Aguiar, P Fernandes, B Bourguignon, MS Khots, DL Massart, and R Phan-Than-Luu. 1995. D-optimal designs. *Chemometrics and Intelligent Laboratory Systems* 30(2):199–210.

Ahmad, Subutai, Gerald Tesauro, and Yu He. 1990. Asymptotic convergence of back-propagation: numerical experiments. In *Advances in neural information processing systems*, 606–613.

Aisen, Paul S. 2011. Clinical trial methodologies for disease-modifying therapeutic approaches. *Neurobiology of aging* 32:S64–S66.

Albert, Marilyn S, Steven T DeKosky, Dennis Dickson, Bruno Dubois, Howard H Feldman, Nick C Fox, Anthony Gamst, David M Holtzman, William J Jagust, Ronald C Petersen, et al. 2011a. The diagnosis of mild cognitive impairment due to alzheimer's disease: Recommendations from the national institute on aging-alzheimer's association workgroups on diagnostic guidelines for alzheimer's disease. *Alzheimer's & dementia* 7(3):270–279.

Albert, Marilyn S, Steven T DeKosky, Dennis Dickson, Bruno Dubois, Howard H Feldman, Nick C Fox, Anthony Gamst, David M Holtzman, William J Jagust, Ronald C Petersen, et al. 2011b. The diagnosis of mild cognitive impairment due to alzheimer's disease: Recommendations from the national institute on aging-alzheimer's association workgroups on diagnostic guidelines for alzheimer's disease. *Alzheimer's & dementia* 7(3):270–279.

Andrychowicz, Marcin, Misha Denil, Sergio Gomez, Matthew W Hoffman, David Pfau, Tom Schaul, and Nando de Freitas. 2016. Learning to learn by gradient descent by gradient descent. *arXiv preprint arXiv:1606.04474*.

Arlot, Sylvain, Alain Celisse, et al. 2010. A survey of cross-validation procedures for model selection. *Statistics surveys* 4:40–79.

Arndt, S., T. Cizadlo, N.C. Andreasen, D. Heckel, S. Gold, and D.S. O'Leary. 1996. Tests for comparing images based on randomization and permutation methods. *Journal of Cerebral Blood Flow and Metaholism* 16:1271–1279.

Arora, Sanjeev, Aditya Bhaskara, Rong Ge, and Tengyu Ma. 2014. Provable bounds for learning some deep representations. In *Icml*, 584–592.

Arora, Sanjeev, Yingyu Liang, and Tengyu Ma. 2015. Why are deep nets reversible: A simple theory, with implications for training. *arXiv preprint arXiv:1511.05653*.

Ashburner, J., and K. J. Friston. 2000. Voxel-based morphometry–the methods. *NeuroImage* 11(6):805–821.

Ashburner, J., and K. J. Friston. 2001. Why voxel-based morphometry should be used. *NeuroImage* 14(6):1238–1243.

Ashburner, John. 2010. Vbm tutorial.

Ashburner, John, Gareth Barnes, C Chen, Jean Daunizeau, Guillaume Flandin, Karl Friston, et al. Spm8 manual.

Association, Alzheimer's. 2015. 2015 alzheimer's disease facts and figures. *Alzheimer's & dementia: the journal of the Alzheimer's Association* 11(3):332.

Bach, Francis. 2014. Breaking the curse of dimensionality with convex neural networks. *arXiv preprint arXiv:1412.8690*.

Badrinarayanan, Vijay, Alex Kendall, and Roberto Cipolla. 2015. Segnet: A deep convolutional encoder-decoder architecture for image segmentation. *arXiv preprint arXiv:1511.00561*.

Baldi, Pierre, and Peter Sadowski. 2014. The dropout learning algorithm. *Artificial intelligence* 210:78–122.

Baldi, Pierre, Peter Sadowski, and Daniel Whiteson. 2014. Searching for exotic particles in high-energy physics with deep learning. *Nature communications* 5.

Balzano, L., R. Nowak, and B. Recht. 2007. Online identification and tracking of subspaces from highly incomplete information. *Arxiv Preprint*. Arxiv:1006.4046.

Balzano, L., R. Nowak, and B. Recht. 2010. Online identification and tracking of subspaces from highly incomplete information. 704–711.

Bateman, Randall J, Chengjie Xiong, Tammie LS Benzinger, Anne M Fagan, Alison Goate, Nick C Fox, Daniel S Marcus, Nigel J Cairns, Xianyun Xie, Tyler M Blazey, et al. 2012. Clinical and biomarker changes in dominantly inherited alzheimer's disease. *New England Journal of Medicine* 367(9):795–804.

Becker, Sue, and Yann Le Cun. 1988. Improving the convergence of back-propagation learning with second order methods. In *Proceedings of the 1988 connectionist models summer school*, 29–37. San Matteo, CA: Morgan Kaufmann.

Bellido, I, and Emile Fiesler. 1993. Do backpropagation trained neural networks have normal weight distributions? In *IcannâĽ™93*, 772–775. Springer.

Ben-David, Shai, John Blitzer, Koby Crammer, Alex Kulesza, Fernando Pereira, and Jennifer Wortman Vaughan. 2010. A theory of learning from different domains. *Machine learning* 79(1-2):151–175.

Benaych-Georges, F., and R. R. Nadakuditi. 2011a. The eigenvalues and eigenvectors of finite, low rank perturbations of large random matrices. *Advances in Mathematics* 227(1):494–521.

Benaych-Georges, Florent, and Raj Rao Nadakuditi. 2011b. The eigenvalues and eigenvectors of finite, low rank perturbations of large random matrices. *Advances in Mathematics* 227(1):494–521.

Bengio, Y. 2009a. Learning deep architectures for AI. *Foundations and trends in Machine Learning* 2(1):1–127.

Bengio, Y., P. Lamblin, D. Popovici, H. Larochelle, et al. 2007. Greedy layer-wise training of deep networks. *Advances in Neural information processing systems* 19:153.

Bengio, Yoshua. 2009b. Learning deep architectures for AI. *Foundations and Trends in Machine Learning* 2:1–127.

Bengio, Yoshua. 2012. Practical recommendations for gradient-based training of deep architectures. In *Neural networks: Tricks of the trade*, 437–478. Springer.

Bengio, Yoshua, Aaron Courville, and Pascal Vincent. 2013a. Representation learning: A review and new perspectives. *Pattern Analysis and Machine Intelligence, IEEE Transactions on* 35(8):1798–1828.

Bengio, Yoshua, Aaron Courville, and Pascal Vincent. 2013b. Representation learning: A review and new perspectives.

Bengio, Yoshua, Ian J Goodfellow, and Aaron Courville. 2015. Deep learning. *An MIT Press book in preparation. Draft chapters available at http://www. iro. umontreal. ca/ bengioy/dlbook*.

Bennett, James, and Stan Lanning. 2007. The netflix prize. In *Proceedings of kdd cup and workshop*, vol. 2007, 35.

Bianchini, M., and F. Scarselli. 2014. On the complexity of neural network classifiers: A comparison between shallow and deep architectures. In *Neural networks and learning systems, ieee transactions on*, 1553–1565.

Bishop, Chris M. 1995. Training with noise is equivalent to tikhonov regularization. *Neural computation* 7(1):108–116.

Bishop, Christopher M. 2006. *Pattern Recognition and Machine Learning*. Springer.

Bland, J. M., and D. G. Altman. 1995. Multiple significance tests: the bonferroni method. *British Medical Journal* 310(6973):170.

Blennow, Kaj, Harald Hampel, Michael Weiner, and Henrik Zetterberg. 2010. Cerebrospinal fluid and plasma biomarkers in alzheimer disease. *Nature Reviews Neurology* 6(3):131–144.

Blundell, Charles, Julien Cornebise, Koray Kavukcuoglu, and Daan Wierstra. 2015. Weight uncertainty in neural networks. *arXiv preprint arXiv:1505.05424*.

Bottou, Léon. 1991. Stochastic gradient learning in neural networks. *Proceedings of Neuro-Nımes* 91(8).

Bottou, Léon. 2010. Large-scale machine learning with stochastic gradient descent. In *Proceedings of compstat'2010*, 177–186. Springer.

Bourgon, Richard, Robert Gentleman, and Wolfgang Huber. 2010. Independent filtering increases detection power for high-throughput experiments. *Proceedings of the National Academy of Sciences* 107(21):9546–9551.

Boutsidis, Christos, Petros Drineas, and Michael W Mahoney. 2009. Unsupervised feature selection for the k-means clustering problem. In *Advances in neural information processing systems*, 153–161.

Bower, James M, and Hamid Bolouri. 2001. *Computational modeling of genetic and biochemical networks*. MIT press.

Box, Joan Fisher. 1980. Ra fisher and the design of experiments, 1922–1926. *The American Statistician* 34(1):1–7.

Breiman, Leo, et al. 2001. Statistical modeling: The two cultures (with comments and a rejoinder by the author). *Statistical science* 16(3):199–231.

Brown, W Michael, Shawn Martin, Sara N Pollock, Evangelos A Coutsias, and Jean-Paul Watson. 2008. Algorithmic dimensionality reduction for molecular structure analysis. *The Journal of chemical physics* 129(6):064118.

Bruna, Joan, and Stéphane Mallat. 2013. Invariant scattering convolution networks. *IEEE transactions on pattern analysis and machine intelligence* 35(8):1872–1886.

Bull, JP. 1959. The historical development of clinical therapeutic trials. *Journal of chronic diseases* 10(3):218–248.

Candès, E.J, and T. Tao. 2010. The power of convex relaxation: Near-optimal matrix completion. *IEEE Trans. Inf. Theor.* 56(5):2053–2080.

Candes, Emmanuel J, and David L Donoho. 2000. Curvelets: A surprisingly effective nonadaptive representation for objects with edges. Tech. Rep., DTIC Document.

Candès, Emmanuel J, and Benjamin Recht. 2009. Exact matrix completion via convex optimization. *Foundations of Computational mathematics* 9(6):717.

Cao, Xudong, Yichen Wei, Fang Wen, and Jian Sun. 2014. Face alignment by explicit shape regression. *International Journal of Computer Vision* 107(2):177–190.

Carmena, Jose M, Mikhail A Lebedev, Roy E Crist, Joseph E O'Doherty, David M Santucci, Dragan F Dimitrov, Parag G Patil, Craig S Henriquez, and Miguel AL Nicolelis. 2003. Learning to control a brain–machine interface for reaching and grasping by primates. *PLoS biol* 1(2):e42.

Castillo, Enrique, Bertha Guijarro-Berdiñas, Oscar Fontenla-Romero, and Amparo Alonso-Betanzos. 2006. A very fast learning method for neural networks based on sensitivity analysis. *The Journal of Machine Learning Research* 7:1159–1182.

Chandrasekaran, S, M Gu, and W Lyons. 2005. A fast adaptive solver for hierarchically semiseparable representations. *Calcolo* 42(3-4):171–185.

Chandrasekaran, Venkat, Sujay Sanghavi, Pablo A Parrilo, and Alan S Willsky. 2011. Rank-sparsity incoherence for matrix decomposition. *SIAM Journal on Optimization* 21(2):572–596.

Chatfield, Ken, Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. 2014. Return of the devil in the details: Delving deep into convolutional nets. *arXiv preprint arXiv:1405.3531*.

Chen, Scott Shaobing, David L Donoho, and Michael A Saunders. 2001. Atomic decomposition by basis pursuit. *SIAM review* 43(1):129–159.

Cheriyadat, Anil M, and Richard J Radke. 2009. Non-negative matrix factorization of partial track data for motion segmentation. In *2009 ieee 12th international conference on computer vision*, 865–872. IEEE.

Chetelat, G, B Desgranges, V De La Sayette, F Viader, F Eustache, and J-C Baron. 2003. Mild cognitive impairment: Can fdg-pet predict who is to rapidly convert to alzheimer's disease? *Neurology* 60(8):1374–1377.

Cheverud, J. M. 2001a. A simple correction for multiple comparisons in interval mapping genome scans. *Heredity* 87(1):52–58.

Cheverud, James M. 2001b. A simple correction for multiple comparisons in interval mapping genome scans. *Heredity* 87(1):52–58.

Ciregan, Dan, Ueli Meier, and Jürgen Schmidhuber. 2012. Multi-column deep neural networks for image classification. In *Computer vision and pattern recognition (cvpr), 2012 ieee conference on*, 3642–3649. IEEE.

Clarke, S., and P. Hall. 2009. Robustness of multiple testing procedures against dependence. *The Annals of Statistics* 332–358.

Coifman, Ronald R, and Mauro Maggioni. 2006. Diffusion wavelets. *Applied and Computational Harmonic Analysis* 21(1):53–94.

Cook, Thomas D, and David L DeMets. 2007. *Introduction to statistical methods for clinical trials*. CRC Press.

Corder, EH, AM Saunders, WJ Strittmatter, DE Schmechel, PC Gaskell, GWet al Small, AD Roses, JL Haines, and M Al Pericak-Vance. 1993. Gene dose of apolipoprotein e type 4 allele and the risk of alzheimer's disease in late onset families. *Science* 261(5123): 921–923.

Corouge, Isabelle, Sylvain Gouttard, and Guido Gerig. 2004. Towards a shape model of white matter fiber bundles using diffusion tensor mri. In *Biomedical imaging: Nano to macro, 2004. ieee international symposium on*, 344–347. IEEE.

Dahl, David B, and Michael A Newton. 2007. Multiple hypothesis testing by clustering treatment effects. *Journal of the American Statistical Association* 102(478):517–526.

Dahl, G., D. Yu, L. Deng, and A. Acero. 2011. Large vocabulary continuous speech recognition with context-dependent DBN-HMMs. In *Acoustics, speech and signal processing (icassp), proceedings of*, 4688–4691.

Dahl, George E, Tara N Sainath, and Geoffrey E Hinton. 2013. Improving deep neural networks for LVCSR using rectified linear units and dropout. In *Acoustics, speech and signal processing (icassp), 2013 ieee international conference on*, 8609–8613. IEEE.

Danaher, Patrick, Pei Wang, and Daniela M Witten. 2014. The joint graphical lasso for inverse covariance estimation across multiple classes. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* 76(2):373–397.

Dauphin, Yann N, Razvan Pascanu, Caglar Gulcehre, Kyunghyun Cho, Surya Ganguli, and Yoshua Bengio. 2014. Identifying and attacking the saddle point problem in high-dimensional non-convex optimization. In *Advances in neural information processing systems*, 2933–2941.

Davatzikos, Christos, Priyanka Bhatt, Leslie M Shaw, Kayhan N Batmanghelich, and John Q Trojanowski. 2011. Prediction of MCI to AD conversion, via MRI, CSF biomarkers, and pattern classification. *Neurobiology of aging* 32:2322–e19.

DeJong, Kenneth A, and William M Spears. 1990. Learning concept classification rules using genetic algorithms. Tech. Rep., DTIC Document.

DerSimonian, Rebecca, and Nan Laird. 1986. Meta-analysis in clinical trials. *Controlled clinical trials* 7(3):177–188.

Dietterich, Thomas G. 1997. Machine-learning research. *AI magazine* 18(4):97–136.

Dietterich, Thomas G. 2000. Ensemble methods in Machine learning. *Multiple Classificer Systems* 1–15.

Dong, Yinpeng, Hang Su, Jun Zhu, and Bo Zhang. 2017. Improving interpretability of deep neural networks with semantic information. *arXiv preprint arXiv:1703.04096*.

Donner, Allan. 1984. Approaches to sample size estimation in the design of clinical trialsâĽ"a review. *Statistics in medicine* 3(3):199–214.

Dosovitskiy, Alexey, and Thomas Brox. 2015. Inverting visual representations with convolutional networks. *arXiv preprint arXiv:1506.02753*.

Du, Simon S, Chi Jin, Jason D Lee, Michael I Jordan, Aarti Singh, and Barnabas Poczos. 2017. Gradient descent can take exponential time to escape saddle points. In *Advances in neural information processing systems*, 1067–1077.

Dubois, Bruno, Howard H Feldman, Claudia Jacova, Steven T DeKosky, Pascale Barberger-Gateau, Jeffrey Cummings, André Delacourte, Douglas Galasko, Serge Gauthier, Gregory Jicha, et al. 2007. Research criteria for the diagnosis of alzheimer's disease: revising the nincds–adrda criteria. *The Lancet Neurology* 6(8):734–746.

Dubois, Bruno, Harald Hampel, Howard H Feldman, Philip Scheltens, Paul Aisen, Sandrine Andrieu, Hovagim Bakardjian, Habib Benali, Lars Bertram, Kaj Blennow, et al. 2016. Preclinical alzheimer's disease: definition, natural history, and diagnostic criteria. *Alzheimer's & Dementia* 12(3):292–323.

Dwass, M. 1957. Modified randomization tests for nonparametric hypotheses. *The Annals of Mathematical Statistics* 28(1):181–187.

Edelman, A. 1988. Eigenvalues and condition numbers of random matrices. *SIAM Journal on Matrix Analysis and Applications* 9(4):543–560.

Edgington, E.S. 1969a. Approximate randomization tests. *The Journal of Psychology* 72(2):143–149.

Edgington, E.S. 1969b. Statistical inference: The distribution-free approach.

Edgington, Eugene, and Patrick Onghena. 2007. *Randomization tests*. CRC Press.

Edison, P, HA Archer, R Hinz, A Hammers, N Pavese, YF Tai, G Hotton, D Cutler, N Fox, A Kennedy, et al. 2007. Amyloid, hypometabolism, and cognition in alzheimer disease an [11c] pib and [18f] fdg pet study. *Neurology* 68(7):501–508.

Edmonds, Emily C, Lisa Delano-Wood, Lindsay R Clark, Amy J Jak, Daniel A Nation, Carrie R McDonald, David J Libon, Rhoda Au, Douglas Galasko, David P Salmon, et al. 2015. Susceptibility of the conventional criteria for mild cognitive impairment to false-positive diagnostic errors. *Alzheimer's & Dementia* 11(4):415–424.

Efron, Bradley, and Robert Tibshirani. 2007. On testing the significance of sets of genes. *The annals of applied statistics* 107–129.

Eklund, Anders, Mats Andersson, and Hans Knutsson. 2011. Fast random permutation tests enable objective evaluation of methods for single-subject fmri analysis. *International journal of biomedical imaging* 2011.

Erhan, D., P. Manzagol, Y. Bengio, S. Bengio, and P. Vincent. 2009. The difficulty of training deep architectures and the effect of unsupervised pre-training. In *Proceedings of the international conference on artificial intelligence and statistics*, 153–160.

Erhan, Dumitru, Yoshua Bengio, Aaron Courville, Pierre-Antoine Manzagol, Pascal Vincent, and Samy Bengio. 2010a. Why does unsupervised pre-training help deep learning? *JMLR* 11:625–660.

Erhan, Dumitru, Yoshua Bengio, Aaron Courville, Pierre-Antoine Manzagol, Pascal Vincent, and Samy Bengio. 2010b. Why does unsupervised pre-training help deep learning? *Journal of Machine Learning Research* 11(Feb):625–660.

Erhan, Dumitru, Aaron Courville, and Yoshua Bengio. 2010c. Understanding representations learned in deep architectures. *Department d'Informatique et Recherche Operationnelle, University of Montreal, QC, Canada, Tech. Rep* 1355.

Escudero, Javier, John P Zajicek, and Emmanuel Ifeachor. 2011. Machine learning classification of mri features of alzheimer's disease and mild cognitive impairment subjects to reduce the sample size in clinical trials. In *Engineering in medicine and biology society, embc, 2011 annual international conference of the ieee*, 7957–7960. IEEE.

Fazel, Maryam, Haitham Hindi, and S Boyd. 2004. Rank minimization and applications in system theory. In *American control conference, 2004. proceedings of the 2004*, vol. 4, 3273–3278. IEEE.

Fei-Fei, Li, Rob Fergus, and Pietro Perona. 2007. Learning generative visual models from few training examples: An incremental bayesian approach tested on 101 object categories. *Computer Vision and Image Understanding* 106(1):59–70.

Fennema-Notestine, Christine, Donald J Hagler, Linda K McEvoy, Adam S Fleisher, Elaine H Wu, David S Karow, and Anders M Dale. 2009. Structural mri biomarkers for preclinical and mild alzheimer's disease. *Human brain mapping* 30(10):3238–3253.

Ferri, Cleusa P, Martin Prince, Carol Brayne, Henry Brodaty, Laura Fratiglioni, Mary Ganguli, Kathleen Hall, Kazuo Hasegawa, Hugh Hendrie, Yueqin Huang, et al. 2006. Global prevalence of dementia: a delphi consensus study. *The lancet* 366(9503):2112–2117.

Feurer, Matthias, Jost Tobias Springenberg, and Frank Hutter. 2015. Initializing bayesian hyperparameter optimization via meta-learning. In *Aaai*, 1128–1135.

Finner, H., and V. Gontscharuk. 2009. Controlling the familywise error rate with plug-in estimator for the proportion of true null hypotheses. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* 71(5):1031–1048.

Fisher, RA. 1934. Biological monographs and manuals vol-5.

Friedman, Lawrence M, Curt D Furberg, and David L DeMets. 2010. Fundamentals of clinical trials. Springer.

Frisoni, Giovanni B, Nick C Fox, Clifford R Jack, Philip Scheltens, and Paul M Thompson. 2010. The clinical use of structural mri in alzheimer disease. *Nature Reviews Neurology* 6(2):67–77.

Friston, Karl J, Daniel E Glaser, Richard NA Henson, S Kiebel, Christophe Phillips, and John Ashburner. 2002. Classical and bayesian inference in neuroimaging: applications. *Neuroimage* 16(2):484–512.

Friston, Karl J, Andrew P Holmes, Keith J Worsley, J-P Poline, Chris D Frith, and Richard SJ Frackowiak. 1994. Statistical parametric maps in functional imaging: a general linear approach. *Human brain mapping* 2(4):189–210.

FSL. 2012. Fmrib software library (fsl). Available online at http://fsl.fmrib.ox.ac.uk/fsl/fslwiki/.

Gaonkar, B., and C. Davatzikos. 2013a. Analytic estimation of statistical significance maps for support vector machine based multi-variate image analysis and classification. *NeuroImage* 78:270–283.

Gaonkar, Bilwaj, and Christos Davatzikos. 2013b. Analytic estimation of statistical significance maps for support vector machine based multi-variate image analysis and classification. *NeuroImage* 78:270–283.

García, S., A. Fernández, J. Luengo, and F. Herrera. 2010. Advanced nonparametric tests for multiple comparisons in the design of experiments in computational intelligence and data mining: Experimental analysis of power. *Information Sciences* 180(10):2044–2064.

Gavish, Matan, and Ronald R Coifman. 2012. Sampling, denoising and compression of matrices by coherent matrix organization. *Applied and Computational Harmonic Analysis* 33(3):354–369.

Ge, Y., S. Dudoit, and T. P. Speed. 2003. Resampling-based multiple testing for microarray data analysis. *Test* 12(1):1–77.

Gerber, Samuel, Tolga Tasdizen, Sarang Joshi, and Ross Whitaker. 2009. On the manifold structure of the space of brain images. In *International conference on medical image computing and computer-assisted intervention*, 305–312. Springer.

Ghadimi, S., and G. Lan. 2013. Stochastic first-and zeroth-order methods for nonconvex stochastic programming. *SIAM Journal on Optimization* 23(4):2341–2368.

Ghadimi, Saeed, Guanghui Lan, and Hongchao Zhang. 2016. Mini-batch stochastic approximation methods for nonconvex stochastic composite optimization. *Mathematical Programming* 155(1-2):267–305.

Goh, Hanlin, Nicolas Thome, Matthieu Cord, and Joo-Hwee Lim. 2013. Top-down regularization of deep belief networks. In *Advances in neural information processing systems*, 1878–1886.

Goodfellow, Ian J, David Warde-Farley, Mehdi Mirza, Aaron C Courville, and Yoshua Bengio. 2013. Maxout networks. *ICML (3)* 28:1319–1327.

Grill, Joshua D, Lijie Di, Po H Lu, Cathy Lee, John Ringman, Liana G Apostolova, et al. 2013. Estimating sample sizes for predementia Alzheimer's trials based on the Alzheimer's Disease Neuroimaging Initiative. *Neurobiology of aging* 34:62–72.

Grill, Joshua D, and Sarah E Monsell. 2014. Choosing alzheimer's disease prevention clinical trial populations. *Neurobiology of aging* 35(3):466–471.

Groves, Peter, Basel Kayyali, David Knott, and Steve Van Kuiken. 2016. The 'big data' revolution in healthcare: Accelerating value and innovation.

Gupta, Ashish, Murat Ayhan, and Anthony Maida. 2013. Natural image bases to represent neuroimaging data. In *Proceedings of the 30th icml*, 987–994.

Gutierrez-Barragan, Felipe, Vamsi K Ithapu, Chris Hinrichs, Camille Maumet, Sterling C Johnson, Thomas E Nichols, Vikas Singh, Alzheimer's Disease Neuroimaging Initiative, et al. 2017. Accelerating permutation testing in voxel-wise analysis through subspace tracking: A new plugin for snpm. *NeuroImage* 159:79–98.

Haeffele, Benjamin D, and René Vidal. 2017. Global optimality in neural network training. In *Proceedings of the ieee conference on computer vision and pattern recognition*, 7331–7339.

Haidich, AB. 2011. Meta-analysis in medical research. *Hippokratia* 14(1):29–37.

Hall, Nancy S. 2007. Ra fisher and his advocacy of randomization. *Journal of the History of Biology* 40(2):295.

Hardt, Moritz, Benjamin Recht, and Yoram Singer. 2015. Train faster, generalize better: Stability of stochastic gradient descent. *arXiv preprint arXiv:1509.01240*.

He, J., L. Balzano, and A. Szlam. 2012a. Incremental gradient on the grassmannian for online foreground and background separation in subsampled video. In *Computer vision and pattern recognition (cvpr), 2012 ieee conference on*, 1568–1575.

He, Jun, Laura Balzano, and Arthur Szlam. 2012b. Incremental gradient on the grassmannian for online foreground and background separation in subsampled video. In *Computer vision and pattern recognition (cvpr), 2012 ieee conference on*, 1568–1575. IEEE.

He, Kaiming, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the ieee conference on computer vision and pattern recognition*, 770–778.

He, Yong, Liang Wang, Yufeng Zang, Lixia Tian, Xinqing Zhang, Kuncheng Li, and Tianzi Jiang. 2007. Regional coherence changes in the early stages of alzheimer's disease: a combined structural and resting-state functional mri study. *Neuroimage* 35(2):488–500.

Hinrichs, Chris, N.Maritza Dowling, SterlingC. Johnson, and Vikas Singh. 2012a. Mkl-based sample enrichment and customized outcomes enable smaller ad clinical trials. In *Mlini*, vol. 7263 of *LNCS*, 124–131. Springer Berlin Heidelberg.

Hinrichs, Chris, Vamsi K Ithapu, Qinyuan Sun, Sterling C Johnson, and Vikas Singh. 2013. Speeding up permutation testing in neuroimaging. In *Advances in neural information processing systems*, 890–898.

Hinrichs, Chris, Vikas Singh, Jiming Peng, and Sterling Johnson. 2012b. Q-mkl: Matrix-induced regularization in multi-kernel learning with applications to neuroimaging. In *Advances in neural information processing systems*, 1421–1429.

Hinrichs, Chris, Vikas Singh, Guofan Xu, and Sterling C Johnson. 2011a. Predictive markers for AD in a multi-modality framework: an analysis of MCI progression in the ADNI population. *Neuroimage* 55:574–589.

Hinrichs, Chris, Vikas Singh, Guofan Xu, Sterling C Johnson, Alzheimers Disease Neuroimaging Initiative, et al. 2011b. Predictive markers for ad in a multi-modality framework: an analysis of mci progression in the adni population. *Neuroimage* 55(2):574–589.

Hinton, G., and R. Salakhutdinov. 2006. Reducing the dimensionality of data with neural networks. *Science* 313(5786):504–507.

Hinton, Geoffrey. 2010. A practical guide to training restricted boltzmann machines. *Momentum* 9(1):926.

Hinton, Geoffrey, Li Deng, Dong Yu, George E Dahl, Abdel-rahman Mohamed, Navdeep Jaitly, Andrew Senior, Vincent Vanhoucke, Patrick Nguyen, Tara N Sainath, et al. 2012. Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *Signal Processing Magazine, IEEE* 29(6):82–97.

Holmes, A.P., R.C. Blair, J.D. Watson, and I. Ford. 1996. Nonparametric analysis of statistic images from functional mapping experiments. *Journal on Cerebral Blood Flow and Metabolism* 16(1):7–22.

Hoyer, Patrik O. 2004. Non-negative matrix factorization with sparseness constraints. *Journal of machine learning research* 5(Nov):1457–1469.

Hua, Xue, Alex D Leow, Neelroop Parikshak, Suh Lee, Ming-Chang Chiang, Arthur W Toga, Clifford R Jack, Michael W Weiner, Paul M Thompson, Alzheimer's Disease Neuroimaging Initiative, et al. 2008. Tensor-based morphometry as a neuroimaging biomarker for alzheimer's disease: an mri study of 676 ad, mci, and normal subjects. *Neuroimage* 43(3):458–469.

Huang, Gao, Zhuang Liu, Kilian Q Weinberger, and Laurens van der Maaten. 2016. Densely connected convolutional networks. *arXiv preprint arXiv:1608.06993*.

Huang, Wenyong, Chengxuan Qiu, Eva von Strauss, Bengt Winblad, and Laura Fratiglioni. 2004. Apoe genotype, family history of dementia, and alzheimer disease risk: a 6-year follow-up study. *Archives of neurology* 61(12):1930–1934.

Humpel, Christian. 2011. Identifying and validating biomarkers for alzheimer's disease. *Trends in biotechnology* 29(1):26–32.

Hwa Kim, Won, Hyunwoo J Kim, Nagesh Adluru, and Vikas Singh. 2016. Latent variable graphical model selection using harmonic analysis: applications to the human connectome project (hcp). In *Proceedings of the ieee conference on computer vision and pattern recognition*, 2443–2451.

Hyvärinen, Aapo. 2013. Independent component analysis: recent advances. *Phil. Trans. R. Soc. A* 371(1984):20110534.

Ioffe, Sergey, and Christian Szegedy. 2015. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*.

Ithapu, Vamsi K, Risi Kondor, Sterling C Johnson, and Vikas Singh. 2017a. The incremental multiresolution matrix factorization algorithm. In *Proceedings of ieee computer vision and pattern recognition (cvpr)*.

Ithapu, Vamsi K, Sathya Ravi, and Vikas Singh. 2015a. On the interplay of network structure and gradient convergence in deep learning. *arXiv preprint arXiv:1511.05297*.

Ithapu, Vamsi K, Sathya N Ravi, and Vikas Singh. 2016. On the interplay of network structure and gradient convergence in deep learning. In *Communication, control, and computing (allerton), 2016 54th annual allerton conference on*, 488–495. IEEE.

Ithapu, Vamsi K, Sathya N Ravi, and Vikas Singh. 2017b. On architectural choices in deep learning: From network structure to gradient convergence and parameter estimation. *arXiv preprint arXiv:1702.08670*.

Ithapu, Vamsi K, Vikas Singh, Ozioma Okonkwo, Richard Chappell, and Sterling C Johnson. 2014a. A predictive multimodal imaging marker for efficient sample enrichment in ad clinical trials. *Alzheimer's & Dementia: The Journal of the Alzheimer's Association* 10(4):P889–P890.

Ithapu, Vamsi K, Vikas Singh, Ozioma Okonkwo, and Sterling C Johnson. 2014b. Randomized denoising autoencoders for smaller and efficient imaging based ad clinical trials. In *International conference on medical image computing and computer-assisted intervention*, 470–478. Springer.

Ithapu, Vamsi K, Vikas Singh, Ozioma C Okonkwo, Richard J Chappell, N Maritza Dowling, and Sterling C Johnson. 2015b. Imaging-based enrichment criteria using deep learning algorithms for efficient clinical trials in mild cognitive impairment. *Alzheimer's & dementia: the journal of the Alzheimer's Association* 11(12):1489–1499.

Jack, Clifford R, Matt A Bernstein, Nick C Fox, Paul Thompson, Gene Alexander, Danielle Harvey, Bret Borowski, Paula J Britson, Jennifer L Whitwell, Chadwick Ward, et al. 2008. The alzheimer's disease neuroimaging initiative (adni): Mri methods. *Journal of Magnetic Resonance Imaging* 27(4):685–691.

Jack, Clifford R, David S Knopman, William J Jagust, Ronald C Petersen, Michael W Weiner, Paul S Aisen, Leslie M Shaw, Prashanthi Vemuri, Heather J Wiste, Stephen D Weigand, et al. 2013. Tracking pathophysiological processes in alzheimer's disease: an updated hypothetical model of dynamic biomarkers. *The Lancet Neurology* 12(2): 207–216.

Jack, Clifford R, Val J Lowe, Stephen D Weigand, Heather J Wiste, Matthew L Senjem, David S Knopman, Maria M Shiung, Jeffrey L Gunter, Bradley F Boeve, Bradley J Kemp, et al. 2009. Serial pib and mri in normal, mild cognitive impairment and alzheimer's disease: implications for sequence of pathological events in alzheimer's disease. *Brain* awp062.

Jahanshad, Neda, Peter V Kochunov, Emma Sprooten, René C Mandl, Thomas E Nichols, Laura Almasy, John Blangero, Rachel M Brouwer, Joanne E Curran, Greig I de Zubicaray, et al. 2013. Multi-site genetic analysis of diffusion images and voxelwise heritability analysis: A pilot project of the enigma–dti working group. *Neuroimage* 81: 455–469.

Jain, Prateek, and Praneeth Netrapalli. 2015. Fast exact matrix completion with finite samples. In *Conference on learning theory*, 1007–1034.

Janzamin, Majid, Hanie Sedghi, and Anima Anandkumar. 2015. Beating the perils of non-convexity: Guaranteed training of neural networks using tensor methods. *CoRR abs/1506.08473*.

Jelic, Vesna, Miia Kivipelto, and Bengt Winblad. 2006. Clinical trials in mild cognitive impairment: lessons for the future. *Journal of Neurology, Neurosurgery & Psychiatry* 77(4): 429–438.

Ji, Hui, Chaoqiang Liu, Zuowei Shen, and Yuhong Xu. 2010. Robust video denoising using low rank matrix completion. In *Cvpr*, 1791–1798. Citeseer.

Jockel, K.-H. 1984. *Computational aspects of monte carlo tests*. Heidelberg: Physica-Verlag HD.

Johnson, Sterling C, Rebecca L Koscik, Erin M Jonaitis, Lindsay R Clark, Kimberly D Mueller, Sara E Berman, Barbara B Bendlin, Corinne D Engelman, Ozioma C Okonkwo, Kirk J Hogan, et al. 2017. The wisconsin registry for alzheimer's prevention: A review of findings and current directions. *Alzheimer's & Dementia: Diagnosis, Assessment & Disease Monitoring*.

Jozefowicz, Rafal, Wojciech Zaremba, and Ilya Sutskever. 2015. An empirical exploration of recurrent network architectures. In *Proceedings of the 32nd international conference on machine learning (icml-15)*, 2342–2350.

Katehakis, Michael N, and Arthur F Veinott Jr. 1987. The multi-armed bandit problem: decomposition and computation. *Mathematics of Operations Research* 12(2):262–268.

Kavukcuoglu, K., P. Sermanet, Y. Boureau, K. Gregor, M. Mathieu, and Y. LeCun. 2010. Learning convolutional feature hierarchies for visual recognition. In *Advances in neural information processing systems*, vol. 1, 5.

Kay, Stephen. 1993. *Fundamentals of Statistical Signal Processing: Estimation Theory*. Prentice Hall.

Keskar, Nitish Shirish, Dheevatsa Mudigere, Jorge Nocedal, Mikhail Smelyanskiy, and Ping Tak Peter Tang. 2016. On large-batch training for deep learning: Generalization gap and sharp minima. *arXiv preprint arXiv:1609.04836*.

Killiany, Ronald J, Teresa Gomez-Isla, Mark Moss, Ron Kikinis, Tamas Sandor, Ferenc Jolesz, Rudolph Tanzi, Kenneth Jones, Bradley T Hyman, and Marilyn S Albert. 2000. Use of structural magnetic resonance imaging to predict who will get alzheimer's disease. *Annals of neurology* 47(4):430–439.

Kim, Won Hwa, Vikas Singh, Moo K Chung, Chris Hinrichs, Deepti Pachauri, Ozioma C Okonkwo, Sterling C Johnson, Alzheimer's Disease Neuroimaging Initiative, et al. 2014. Multi-resolutional shape features via non-euclidean wavelets: Applications to statistical analysis of cortical thickness. *NeuroImage* 93:107–123.

K.J. Friston, J-B. Polline P.J. Grasby S.C.R. Williams R.S.J. Frackowiak, A.P. Holmes. 1995. Statistical parametric maps in functional imaging: A general linear approach. *Human Brain Mapping* 2:189–210.

Klonoff, David C, John B Buse, Loretta L Nielsen, Xuesong Guan, Christopher L Bowlus, John H Holcombe, Matthew E Wintle, and David G Maggs. 2008. Exenatide effects on diabetes, obesity, cardiovascular risk factors and hepatic biomarkers in patients with type 2 diabetes treated for at least 3 years. *Current medical research and opinion* 24(1): 275–286.

Klöppel, Stefan, Cynthia M Stonnington, Carlton Chu, Bogdan Draganski, Rachael I Scahill, Jonathan D Rohrer, Nick C Fox, Clifford R Jack, John Ashburner, and Richard SJ Frackowiak. 2008. Automatic classification of mr scans in alzheimer's disease. *Brain* 131(3):681–689.

Klunk, William E, Henry Engler, Agneta Nordberg, Yanming Wang, Gunnar Blomqvist, Daniel P Holt, Mats Bergström, Irina Savitcheva, Guo-Feng Huang, Sergio Estrada, et al. 2004. Imaging brain amyloid in alzheimer's disease with pittsburgh compound-b. *Annals of neurology* 55(3):306–319.

Klunk, William E, Robert A Koeppe, Julie C Price, Tammie L Benzinger, Michael D Devous, William J Jagust, Keith A Johnson, Chester A Mathis, Davneet Minhas, Michael J Pontecorvo, et al. 2015. The centiloid project: standardizing quantitative amyloid plaque estimation by pet. *Alzheimer's & Dementia* 11(1):1–15.

Knijnenburg, Theo A, Lodewyk FA Wessels, Marcel JT Reinders, and Ilya Shmulevich. 2009. Fewer permutations, more accurate p-values. *Bioinformatics* 25(12):i161–i168.

Kohannim, Omid, Xue Hua, Derrek P Hibar, Suh Lee, Yi-Yu Chou, Arthur W Toga, Clifford R Jack Jr, Michael W Weiner, and Paul M Thompson. 2010. Boosting power for clinical trials using classifiers based on multiple biomarkers. *Neurobiology of aging* 31: 1429–1442.

Kondor, Risi, Nedelina Teneva, and Vikas Garg. 2014. Multiresolution matrix factorization. In *Proceedings of the 31st international conference on machine learning (icml-14)*, 1620–1628.

Kondor, Risi, Nedelina Teneva, and Pramod K Mudrakarta. 2015. Parallel mmf: a multiresolution approach to matrix computation. *arXiv preprint arXiv:1507.04396*.

Koren, Yehuda, Robert Bell, and Chris Volinsky. 2009. Matrix factorization techniques for recommender systems. *Computer* 42(8).

Krizhevsky, A., I. Sutskever, and G. Hinton. 2012. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, vol. 1, 4.

Krizhevsky, Alex, and Geoffrey Hinton. 2009. Learning multiple layers of features from tiny images.

Kutyniok, Gitta, et al. 2012. *Shearlets: Multiscale analysis for multivariate data*. Springer Science & Business Media.

Lakkaraju, Himabindu, and Jure Leskovec. 2016. Confusions over time: An interpretable bayesian model to characterize trends in decision making. In *Advances in neural information processing systems*, 3261–3269.

Lakkaraju, Himabindu, and Cynthia Rudin. 2017. Learning cost-effective and interpretable treatment regimes. In *Artificial intelligence and statistics*, 166–175.

Lampos, Vasileios, and Nello Cristianini. 2010. Tracking the flu pandemic by monitoring the social web. In *Cognitive information processing (cip), 2010 2nd international workshop on*, 411–416. IEEE.

Lazar, Mariana, David M Weinstein, Jay S Tsuruda, Khader M Hasan, Konstantinos Arfanakis, M Elizabeth Meyerand, Benham Badie, Howard A Rowley, Victor Haughton, Aaron Field, et al. 2003. White matter tractography using diffusion tensor deflection. *Human brain mapping* 18(4):306–321.

Le Roux, Nicolas, and Yoshua Bengio. 2008. Representational power of restricted boltzmann machines and deep belief networks. *Neural computation* 20(6):1631–1649.

Lebedev, AV, Eric Westman, GJP Van Westen, MG Kramberger, Arvid Lundervold, Dag Aarsland, H Soininen, I Kłoszewska, P Mecocci, M Tsolaki, et al. 2014. Random forest ensembles for detection and prediction of alzheimer's disease with a good between-cohort robustness. *NeuroImage: Clinical* 6:115–125.

LeCun, Yann, Yoshua Bengio, and Geoffrey Hinton. 2015. Deep learning. *Nature* 521: 436–44.

LeCun, Yann, Léon Bottou, Yoshua Bengio, and Patrick Haffner. 1998. Gradient-based learning applied to document recognition. *Proceedings of the IEEE* 86(11):2278–2324.

LeCun, Yann A, Léon Bottou, Genevieve B Orr, and Klaus-Robert Müller. 2012. Efficient backprop. In *Neural networks: Tricks of the trade*, 9–48. Springer.

Lee, Ann B, Boaz Nadler, and Larry Wasserman. 2008. Treelets: an adaptive multi-scale basis for sparse unordered data. *The Annals of Applied Statistics* 435–471.

Lee, H., R. Grosse, R. Ranganath, and A. Ng. 2009. Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations. In *Proceedings of the 26th international conference on machine learning (icml-09)*, 609–616.

Lee, Hyekyoung, Dong Soo Lee, Hyejin Kang, Boong-Nyun Kim, and Moo K Chung. 2011. Sparse brain network recovery under compressed sensing. *IEEE Transactions on Medical Imaging* 30(5):1154–1165.

Leek, J. T., and J. D. Storey. 2008. A general framework for multiple testing dependence. *Proceedings of the National Academy of Sciences* 105(48):18718–18723.

Lemm, Steven, Benjamin Blankertz, Thorsten Dickhaus, and Klaus-Robert Müller. 2011. Introduction to machine learning for brain imaging. *Neuroimage* 56(2):387–399.

Lenz, Ian, Honglak Lee, and Ashutosh Saxena. 2015. Deep learning for detecting robotic grasps. *The International Journal of Robotics Research* 34(4-5):705–724.

Leoutsakos, Jeannie-Marie S, Alexandra L Bartlett, Sarah N Forrester, and Constantine G Lyketsos. 2014. Simulating effects of biomarker enrichment on alzheimer's disease prevention trials: Conceptual framework and example. *Alzheimer's & Dementia* 10(2):152–161.

Letham, Benjamin, Cynthia Rudin, Tyler H McCormick, David Madigan, et al. 2015. Interpretable classifiers using rules and bayesian analysis: Building a better stroke prediction model. *The Annals of Applied Statistics* 9(3):1350–1371.

Li, J., and L. Ji. 2005a. Adjusting multiple testing in multilocus analyses using the eigenvalues of a correlation matrix. *Heredity* 95(3):221–227.

Li, J, and L Ji. 2005b. Adjusting multiple testing in multilocus analyses using the eigenvalues of a correlation matrix. *Heredity* 95(3):221–227.

Li, Lisha, Kevin Jamieson, Giulia DeSalvo, Afshin Rostamizadeh, and Ameet Talwalkar. 2016a. Efficient hyperparameter optimization and infinitely many armed bandits. *arXiv preprint arXiv:1603.06560*.

Li, Lisha, Kevin Jamieson, Giulia DeSalvo, Afshin Rostamizadeh, and Ameet Talwalkar. 2016b. Hyperband: A novel bandit-based approach to hyperparameter optimization. *arXiv preprint arXiv:1603.06560*.

Liang, Yingyu, Maria-Florina F Balcan, Vandana Kanchanapally, and David Woodruff. 2014. Improved distributed principal component analysis. In *Advances in neural information processing systems*, 3113–3121.

Lipton, Zachary C. 2016. The mythos of model interpretability. *arXiv preprint arXiv:1606.03490*.

Livni, R., S. Shalev-Shwartz, and O. Shamir. 2014. On the computational efficiency of training neural networks. In *Advances in neural information processing systems*, 855–863.

Long, Mingsheng, Yue Cao, Jianmin Wang, and Michael I Jordan. 2015. Learning transferable features with deep adaptation networks. In *Icml*, 97–105.

Lopez, Alan D, Colin D Mathers, Majid Ezzati, Dean T Jamison, and Christopher JL Murray. 2006. Global and regional burden of disease and risk factors, 2001: systematic analysis of population health data. *The Lancet* 367(9524):1747–1757.

Lorenzi, Marco, Michael Donohue, Donata Paternico, Cristina Scarpazza, Susanne Ostrowitzki, Olivier Blin, Elaine Irving, GB Frisoni, Alzheimer's Disease Neuroimaging Initiative, et al. 2010. Enrichment through biomarkers in clinical trials of alzheimer's drugs in patients with mild cognitive impairment. *Neurobiology of aging* 31(8):1443–1451.

Lou, Yin, Rich Caruana, and Johannes Gehrke. 2012. Intelligible models for classification and regression. In *Proceedings of the 18th acm sigkdd international conference on knowledge discovery and data mining*, 150–158. ACM.

Lowd, Daniel, and Christopher Meek. 2005. Adversarial learning. In *Proceedings of the eleventh acm sigkdd international conference on knowledge discovery in data mining*, 641–647. ACM.

Lowe, David G. 1999. Object recognition from local scale-invariant features. In *Computer vision, 1999. the proceedings of the seventh ieee international conference on*, vol. 2, 1150–1157. Ieee.

Lusci, Alessandro, Gianluca Pollastri, and Pierre Baldi. 2013. Deep architectures and deep learning in chemoinformatics: the prediction of aqueous solubility for drug-like molecules. *Journal of chemical information and modeling* 53(7):1563–1575.

M. Halber, K. Wienhard G. Pawlik, K. Herholz, and W.D. Heiss. 1997. Performance of a randomization test for single-subject 15o-water pet activation studies. *Journal of Cerebral Blood Flow and Metabolism* 17(10):1033–1039.

Ma, Ping, Michael W Mahoney, and Bin Yu. 2015. A statistical perspective on algorithmic leveraging. *Journal of Machine Learning Research* 16:861–911.

Magoulas, George D., Michael N. Vrahatis, and George S Androulakis. 1999. Improving the convergence of the backpropagation algorithm using learning rate adaptation methods. *Neural Computation* 11(7):1769–1796.

Mallat, Stephane G. 1989. A theory for multiresolution signal decomposition: the wavelet representation. *IEEE transactions on pattern analysis and machine intelligence* 11(7):674–693.

Manjunath, Bangalore S, and Wei-Ying Ma. 1996. Texture features for browsing and retrieval of image data. *IEEE Transactions on pattern analysis and machine intelligence* 18(8):837–842.

Marčenko, A. A., and L. A. Pastur. 1967. Distribution of eigenvalues for some sets of random matrices. *Sbornik: Mathematics* 1(4):457–483.

Matsuoka, Kiyotoshi. 1992. Noise injection into inputs in back-propagation learning. *Systems, Man and Cybernetics, IEEE Transactions on* 22(3):436–440.

Mattsson, Niklas, Ulf Andreasson, Staffan Persson, Maria C Carrillo, Steven Collins, Sonia Chalbot, Neal Cutler, Diane Dufour-Rainfray, Anne M Fagan, Niels HH Heegaard, et al. 2013. Csf biomarker variability in the alzheimer's association quality control program. *Alzheimer's & Dementia* 9(3):251–261.

Mayeux, Richard. 2004. Biomarkers: potential uses and limitations. *NeuroRx* 1(2): 182–188.

McCarty, Catherine A, Rex L Chisholm, Christopher G Chute, Iftikhar J Kullo, Gail P Jarvik, Eric B Larson, Rongling Li, Daniel R Masys, Marylyn D Ritchie, Dan M Roden, et al. 2011. The emerge network: a consortium of biorepositories linked to electronic medical records data for conducting genomic studies. *BMC medical genomics* 4(1):13.

McKhann, Guy M, David S Knopman, Howard Chertkow, Bradley T Hyman, Clifford R Jack, Claudia H Kawas, William E Klunk, Walter J Koroshetz, Jennifer J Manly, Richard Mayeux, et al. 2011. The diagnosis of dementia due to alzheimer's disease: Recommendations from the national institute on aging-alzheimer's association workgroups on diagnostic guidelines for alzheimer's disease. *Alzheimer's & dementia* 7(3):263–269.

Medvedev, Paul, Monica Stanciu, and Michael Brudno. 2009. Computational methods for discovering structural variation with next-generation sequencing. *Nature methods* 6: S13–S20.

Melnik, Roderick. 2015. *Mathematical and computational modeling: with applications in natural and social sciences, engineering, and the arts*. John Wiley & Sons.

Meyer, Yves. 1993. Wavelets-algorithms and applications. *Wavelets-Algorithms and applications Society for Industrial and Applied Mathematics Translation., 142 p.* 1.

Mezzadri, Francesco. 2006. How to generate random matrices from the classical compact groups. *arXiv preprint math-ph/0609050*.

Mills, Edward J, An-Wen Chan, Ping Wu, Andy Vail, Gordon H Guyatt, and Douglas G Altman. 2009. Design, analysis, and presentation of crossover trials. *Trials* 10(1):27.

Minsky, Marvin, and Seymour Papert. 1969. Perceptrons.

Mitchell, Alex J, and Moitaba Shiri-Feshki. 2009. Rate of progression of mild cognitive impairment to dementia–meta-analysis of 41 robust inception cohort studies. *Acta Psychiatrica Scandinavica* 119(4):252–265.

Mnih, Volodymyr, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. 2013. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*.

Montufar, Guido F, Razvan Pascanu, Kyunghyun Cho, and Yoshua Bengio. 2014. On the number of linear regions of deep neural networks. In *Advances in neural information processing systems*, 2924–2932.

Mooney, R. 1999. Relational learning of pattern-match rules for information extraction. In *Proceedings of the sixteenth national conference on artificial intelligence*, vol. 328, 334.

Mordvintsev, Alexander, Christopher Olah, and Mike Tyka. 2015. Inceptionism: Going deeper into neural networks.

Mosconi, Lisa, Valentina Berti, Lidia Glodzik, Alberto Pupi, Susan De Santi, and Mony J de Leon. 2010. Pre-clinical detection of alzheimer's disease using fdg-pet, with or without amyloid imaging. *Journal of Alzheimer's Disease* 20(3):843–854.

Mueller, Susanne G, Michael W Weiner, Leon J Thal, Ronald C Petersen, Clifford R Jack, et al. 2005. Ways toward an early diagnosis in alzheimer's disease: the alzheimer's disease neuroimaging initiative (adni). *Alzheimer's & Dementia* 1(1):55–66.

Murty, Katta G, and Santosh N Kabadi. 1987. Some np-complete problems in quadratic and nonlinear programming. *Mathematical programming* 39(2):117–129.

Nagi, Jawad, Frederick Ducatelle, Gianni A Di Caro, Dan Ciresan, Ueli Meier, Alessandro Giusti, Farrukh Nagi, Jürgen Schmidhuber, and Luca Maria Gambardella. 2011. Max-pooling convolutional neural networks for vision-based hand gesture recognition. In *Signal and image processing applications (icsipa), 2011 ieee international conference on*, 342–347. IEEE.

Nair, Vinod, and Geoffrey E Hinton. 2010. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th international conference on machine learning (icml-10)*, 807–814.

Ngiam, Jiquan, Adam Coates, Ahbik Lahiri, Bobby Prochnow, Quoc V Le, and Andrew Y Ng. 2011. On optimization methods for deep learning. In *Proceedings of the 28th international conference on machine learning (icml-11)*, 265–272.

Nichols, T., and S. Hayasaka. 2003a. Controlling the familywise error rate in functional neuroimaging: a comparative review. *Statistical Methods in Medical Research* 12:419–446.

Nichols, T.E., and S. Hayasaka. 2003b. Controlling the familywise error rate in functional neuroimaging: a comparative review. *Statistical Methods in Medical Research* 12: 419–446.

Nichols, Thomas E., and Andrew P. Holmes. 2002. Nonparametric permutation tests for functional neuroimaging: A primer with examples. *Human Brain Map* 15(1):1–25.

Pang, Tao. 1999. An introduction to computational physics.

Pantazis, D., T. E. Nichols, S. Baillet, and R. M. Leahy. 2005. A comparison of random field theory and permutation methods for the statistical analysis of meg data. *NeuroImage* 25(2):383–394.

Pardalos, Panos M, and Georg Schnitger. 1988. Checking local optimality in constrained quadratic programming is np-hard. *Operations Research Letters* 7(1):33–35.

Patel, Ankit B, Tan Nguyen, and Richard G Baraniuk. 2015. A probabilistic theory of deep learning. *arXiv preprint arXiv:1504.00641*.

Paul, Michael J, and Mark Dredze. 2011. You are what you tweet: Analyzing twitter for public health. *Icwsm* 20:265–272.

Pesaran, M Hashem, and Melvyn Weeks. 2001. Non-nested hypothesis testing: an overview. *A Companion to Theoretical Econometrics* 279–309.

Petersen, Ronald C. 2007. Mild cognitive impairment: current research and clinical implications. In *Seminars in neurology*, vol. 27, 22–31.

Peterson, Joshua C, Joshua T Abbott, and Thomas L Griffiths. 2016. Adapting deep network features to capture psychological representations. *arXiv preprint arXiv:1608.02164*.

Pike, Kerryn E, Greg Savage, Victor L Villemagne, Steven Ng, Simon A Moss, Paul Maruff, Chester A Mathis, William E Klunk, Colin L Masters, and Christopher C Rowe. 2007. β-amyloid imaging and memory in non-demented individuals: evidence for preclinical alzheimer's disease. *Brain* 130(11):2837–2844.

Plis, Sergey M, Devon R Hjelm, Ruslan Salakhutdinov, Elena A Allen, et al. 2014. Deep learning for neuroimaging: a validation study. *Frontiers in neuroscience* 8.

Plis, Sergey M, Devon R Hjelm, Ruslan Salakhutdinov, and Vince D Calhoun. 2013. Deep learning for neuroimaging: a validation study. *arXiv preprint arXiv:1312.5847*.

Rajani, Nazneen, Kate McArdle, and Inderjit S Dhillon. 2015. Parallel k nearest neighbor graph construction using tree-based data structures. In *1st high performance graph mining workshop, sydney, 10 august 2015*.

Rao, Nikhil, Christopher Cox, Rob Nowak, and Timothy T Rogers. 2013. Sparse overlapping sets lasso for multitask learning and its application to fmri analysis. In *Advances in neural information processing systems*, 2202–2210.

Rautaray, Siddharth S, and Anupam Agrawal. 2015. Vision based hand gesture recognition for human computer interaction: a survey. *Artificial Intelligence Review* 43(1): 1–54.

Recht, Benjamin. 2011. A simpler approach to matrix completion. *The Journal of Machine Learning Research* 12:3413–3430.

Recht, Benjamin, Maryam Fazel, and Pablo A Parrilo. 2010. Guaranteed minimum-rank solutions of linear matrix equations via nuclear norm minimization. *SIAM review* 52(3): 471–501.

Rifai, Salah, Xavier Glorot, Yoshua Bengio, and Pascal Vincent. 2011. Adding noise to the input of a model trained with a regularized objective. *arXiv preprint arXiv:1104.3250*.

Rockafellar, R Tyrrell, and Roger J-B Wets. 2009. *Variational analysis*, vol. 317. Springer Science & Business Media.

Romero, Adriana, Nicolas Ballas, Samira Ebrahimi Kahou, Antoine Chassang, Carlo Gatta, and Yoshua Bengio. 2014. Fitnets: Hints for thin deep nets. *arXiv preprint arXiv:1412.6550*.

Rosenblatt, Frank. 1961. Principles of neurodynamics. perceptrons and the theory of brain mechanisms. Tech. Rep., CORNELL AERONAUTICAL LAB INC BUFFALO NY.

Roth, Stefan, and Michael J Black. 2005. Fields of experts: A framework for learning image priors. In *2005 ieee computer society conference on computer vision and pattern recognition (cvpr'05)*, vol. 2, 860–867. IEEE.

Rowe, Christopher C, Uwe Ackerman, William Browne, Rachel Mulligan, Kerryn L Pike, Graeme O'Keefe, Henry Tochon-Danguy, Gordon Chan, Salvatore U Berlangieri, Gareth Jones, et al. 2008. Imaging of amyloid β in alzheimer's disease with 18 f-bay94-9172, a novel pet tracer: proof of mechanism. *The Lancet Neurology* 7(2):129–135.

Rubinstein, Ron, Alfred M Bruckstein, and Michael Elad. 2010. Dictionaries for sparse representation modeling. *Proceedings of the IEEE* 98(6):1045–1057.

Russakovsky, Olga, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. 2015. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision* 115(3):211–252.

Sachidanandam, Ravi, David Weissman, Steven C Schmidt, Jerzy M Kakol, Lincoln D Stein, Gabor Marth, Steve Sherry, James C Mullikin, Beverley J Mortimore, David L Willey, et al. 2001. A map of human genome sequence variation containing 1.42 million single nucleotide polymorphisms. *Nature* 409(6822):928–933.

Sakpal, Tushar Vijay. 2010. Sample size estimation in clinical trial. *Perspectives in clinical research* 1(2):67–69.

Savas, Berkant, Inderjit S Dhillon, et al. 2011. Clustered low rank approximation of graphs in information science applications. In *Sdm*, 164–175. SIAM.

Saxe, A., P. Koh, Z. Chen, M. Bhand, B. Suresh, and A. Ng. 2011. On random weights and unsupervised feature learning. In *Proceedings of the 28th international conference on machine learning (icml-11)*, 1089–1096.

Saxe, Andrew M, James L McClelland, and Surya Ganguli. 2013. Exact solutions to the nonlinear dynamics of learning in deep linear neural networks. *arXiv preprint arXiv:1312.6120*.

Shao, Hongmei, and Gaofeng Zheng. 2011. Convergence analysis of a back-propagation algorithm with adaptive momentum. *Neurocomputing* 74(5):749–752.

Shipp, Margaret A, Ken N Ross, Pablo Tamayo, Andrew P Weng, Jeffery L Kutok, Ricardo CT Aguiar, Michelle Gaasenbeek, Michael Angelo, Michael Reich, Geraldine S Pinkus, et al. 2002. Diffuse large b-cell lymphoma outcome prediction by gene-expression profiling and supervised machine learning. *Nature medicine* 8(1): 68–74.

Shiratori, Takaaki, Yasuyuki Matsushita, Xiaoou Tang, and Sing Bing Kang. 2006. Video completion by motion field transfer. In *Computer vision and pattern recognition, 2006 ieee computer society conference on*, vol. 1, 411–418. IEEE.

Si, Si, Cho-Jui Hsieh, and Inderjit S Dhillon. 2014. Memory efficient kernel approximation. In *Icml*, 701–709.

Simmons, Andrew, Eric Westman, Sebastian Muehlboeck, Patrizia Mecocci, Bruno Vellas, Magda Tsolaki, Iwona Kłoszewska, Lars-Olof Wahlund, Hilkka Soininen, Simon Lovestone, et al. 2011. The addneuromed framework for multi-centre mri assessment of alzheimer's disease: experience from the first 24 months. *International journal of geriatric psychiatry* 26(1):75–82.

Simon, Noah, and Richard Simon. 2013. Adaptive enrichment designs for clinical trials. *Biostatistics* kxt010.

Simonyan, Karen, Andrea Vedaldi, and Andrew Zisserman. 2013. Deep inside convolutional networks: Visualising image classification models and saliency maps. *arXiv preprint arXiv:1312.6034*.

Simonyan, Karen, and Andrew Zisserman. 2014. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*.

Singh, K. D., G. R. Barnes, and A. Hillebrand. 2003. Group imaging of task-related changes in cortical synchronisation using nonparametric permutation testing. *NeuroImage* 19(4):1589–1601.

Smith, Steven Thomas. 2005. Statistical resolution limits and the complexified crame´ r-rao bound. *IEEE Transactions on Signal Processing* 53(5):1597–1609.

Snoek, Jasper, Hugo Larochelle, and Ryan P Adams. 2012. Practical bayesian optimization of machine learning algorithms. In *Advances in neural information processing systems*, 2951–2959.

SnPM. 2013. Statistical non parametric mapping toolbox (snpm). Available online at http://www2.warwick.ac.uk/fac/sci/statistics/staff/academic-research/nichols/software/snpm.

Sperling, Reisa A, Paul S Aisen, Laurel A Beckett, David A Bennett, Suzanne Craft, Anne M Fagan, Takeshi Iwatsubo, Clifford R Jack, Jeffrey Kaye, Thomas J Montine, et al. 2011. Toward defining the preclinical stages of alzheimer's disease: Recommendations from the national institute on aging-alzheimer's association workgroups on diagnostic guidelines for alzheimer's disease. *Alzheimer's & dementia* 7(3):280–292.

Sperling, Reisa A, Dorene M Rentz, Keith A Johnson, Jason Karlawish, Michael Donohue, David P Salmon, and Paul Aisen. 2014. The a4 study: stopping ad before symptoms begin? *Science translational medicine* 6(228):228fs13–228fs13.

SPM. 2012. Statistical parametric mapping toolbox (spm). Available online at http://www.fil.ion.ucl.ac.uk/spm/.

Srivastava, Nitish, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research* 15(1):1929–1958.

Storey, J., and R. Tibshirani. 2003. Statistical significance for genomewide studies. *Proceedings of the National Academy of Sciences* 100(16):9440–9445.

Sturm, Peter, and Bill Triggs. 1996. A factorization based algorithm for multi-image projective structure and motion. In *European conference on computer vision*, 709–720. Springer.

Su, Yu, and Frédéric Jurie. 2012. Improving image classification using semantic attributes. *International journal of computer vision* 100(1):59–77.

Subramanian, Aravind, Pablo Tamayo, Vamsi K Mootha, Sayan Mukherjee, Benjamin L Ebert, Michael A Gillette, Amanda Paulovich, Scott L Pomeroy, Todd R Golub, Eric S Lander, et al. 2005. Gene set enrichment analysis: a knowledge-based approach for interpreting genome-wide expression profiles. *Proceedings of the National Academy of Sciences* 102(43):15545–15550.

Suk, Heung-Il, and Dinggang Shen. 2013. Deep learning-based feature representation for ad/mci classification. In *Miccai*, ed. Kensaku Mori, Ichiro Sakuma, Yoshinobu

Sato, Christian Barillot, and Nassir Navab, vol. 8150 of *LNCS*, 583–590. Springer Berlin Heidelberg.

Sutskever, Ilya, Oriol Vinyals, and Quoc VV Le. 2014. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, 3104–3112.

Swirszcz, Grzegorz, Wojciech Marian Czarnecki, and Razvan Pascanu. 2017. Local minima in training of neural networks. *stat* 1050:17.

Szegedy, Christian, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. 2014. Going deeper with convolutions. *arXiv preprint arXiv:1409.4842*.

Szlezák, N, M Evers, J Wang, and L Pérez. 2014. The role of big data and advanced analytics in drug discovery, development, and commercialization. *Clinical Pharmacology & Therapeutics* 95(5):492–495.

Tan, Aik Choon, and David Gilbert. 2003. Ensemble machine learning on gene expression data for cancer classification.

Tatsuoka, Curtis, Huiyun Tseng, Judith Jaeger, Ferenc Varadi, Mark A Smith, Tomoko Yamada, et al. 2013. Modeling the heterogeneity in risk of progression to Alzheimer's disease across cognitive profiles in mild cognitive impairment. *Alzheimers Res Ther* 5: 14.

Taylor, JE, and KJ Worsley. 2008. Random fields of multivariate test statistics, with applications to shape analysis. *The Annals of Statistics* 36(1):1–27.

Teipel, Stefan J, Christine Born, Michael Ewers, Arun LW Bokde, Maximilian F Reiser, Hans-Jürgen Möller, and Harald Hampel. 2007. Multivariate deformation-based analysis of brain atrophy to predict Alzheimer's disease in mild cognitive impairment. *Neuroimage* 38:13–24.

Teipel, Stefan J, Michel Grothe, Simone Lista, Nicola Toschi, Francesco G Garaci, and Harald Hampel. 2013. Relevance of magnetic resonance imaging for early detection and diagnosis of alzheimer disease. *Medical Clinics of North America* 97(3):399–424.

Teneva, Nedelina, Pramod K Mudrakarta, and Risi Kondor. 2016. Multiresolution matrix compression. In *Proceedings of the 19th international conference on artificial intelligence and statistics*, 1441–1449.

Terry, Robert D, Eliezer Masliah, David P Salmon, Nelson Butters, Richard DeTeresa, Robert Hill, Lawrence A Hansen, and Robert Katzman. 1991. Physical basis of cognitive alterations in alzheimer's disease: synapse loss is the major correlate of cognitive impairment. *Annals of neurology* 30(4):572–580.

Thornton-Wells, Tricia A, Jason H Moore, and Jonathan L Haines. 2004. Genetics, statistics and human disease: analytical retooling for complexity. *TRENDS in Genetics* 20(12):640–647.

Tishby, Naftali, and Noga Zaslavsky. 2015. Deep learning and the information bottleneck principle. In *Information theory workshop (itw), 2015 ieee*, 1–5. IEEE.

De la Torre, Fernando, and Michael J Black. 2001. Robust principal component analysis for computer vision. In *Computer vision, 2001. iccv 2001. proceedings. eighth ieee international conference on*, vol. 1, 362–369. IEEE.

Turk, Matthew, and Alex Pentland. 1991. Eigenfaces for recognition. *Journal of cognitive neuroscience* 3(1):71–86.

Tzourio-Mazoyer, Nathalie, Brigitte Landeau, Dimitri Papathanassiou, Fabrice Crivello, Olivier Etard, Nicolas Delcroix, Bernard Mazoyer, and Marc Joliot. 2002. Automated anatomical labeling of activations in spm using a macroscopic anatomical parcellation of the mni mri single-subject brain. *Neuroimage* 15(1):273–289.

Van Essen, David C, Kamil Ugurbil, E Auerbach, D Barch, TEJ Behrens, R Bucholz, Acer Chang, Liyong Chen, Maurizio Corbetta, Sandra W Curtiss, et al. 2012. The human connectome project: a data acquisition perspective. *Neuroimage* 62(4):2222–2231.

Vemuri, P, HJ Wiste, et al. 2009. MRI and CSF biomarkers in normal, MCI, and AD subjects predicting future clinical change. *Neurology* 73(4):294–301.

Villar, Sofía S, Jack Bowden, and James Wason. 2015. Multi-armed bandit models for the optimal design of clinical trials: benefits and challenges. *Statistical science: a review journal of the Institute of Mathematical Statistics* 30(2):199.

Vincent, P., H. Larochelle, I. Lajoie, Y. Bengio, and P. Manzagol. 2010. Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *The Journal of Machine Learning Research* 9999:3371–3408.

Vogl, Thomas P, JK Mangis, AK Rigler, WT Zink, and DL Alkon. 1988. Accelerating the convergence of the back-propagation method. *Biological cybernetics* 59(4-5):257–263.

Vondrick, Carl, Aditya Khosla, Tomasz Malisiewicz, and Antonio Torralba. 2013. Hoggles: Visualizing object detection features. In *Proceedings of the ieee international conference on computer vision*, 1–8.

Wager, Stefan, Sida Wang, and Percy S Liang. 2013. Dropout training as adaptive regularization. In *Advances in neural information processing systems*, 351–359.

Wan, Li, Matthew Zeiler, Sixin Zhang, Yann L Cun, and Rob Fergus. 2013. Regularization of neural networks using dropconnect. In *Proceedings of the 30th international conference on machine learning (icml-13)*, 1058–1066.

Wang, Haohan, and Bhiksha Raj. 2015. A survey: Time travel in deep learning space: An introduction to deep learning models and how deep learning models evolved from the initial ideas. *arXiv preprint arXiv:1510.04781*.

Wang, Tong, Cynthia Rudin, F Doshi, Yimin Liu, Erica Klampfl, and Perry MacNeille. 2015. Bayesian orâŁ™s of andâŁ™s for interpretable classification with application to context aware recommender systems.

Wei, Tao, Changhu Wang, Rong Rui, and Chang Wen Chen. 2016. Network morphism. *arXiv preprint arXiv:1603.01670*.

Weiner, Michael W, Dallas P Veitch, Paul S Aisen, Laurel A Beckett, Nigel J Cairns, Jesse Cedarbaum, Robert C Green, Danielle Harvey, Clifford R Jack, William Jagust, et al. 2015. 2014 update of the alzheimer's disease neuroimaging initiative: a review of papers published since its inception. *Alzheimer's & Dementia* 11(6):e1–e120.

Weiner, Michael W, Dallas P Veitch, Paul S Aisen, Laurel A Beckett, Nigel J Cairns, Robert C Green, Danielle Harvey, Clifford R Jack, William Jagust, Enchi Liu, et al. 2013. The alzheimer's disease neuroimaging initiative: a review of papers published since its inception. *Alzheimer's & Dementia* 9(5):e111–e194.

Westfall, P. H., and S. S. Young. 1993. *Resampling-based multiple testing: examples and methods for p-value adjustment*, vol. 279. Wiley-Interscience.

Winkler, A. M., G. R. Ridgway, M. A. Webster, S. M. Smith, and T. E. Nichols. 2014. Permutation inference for the general linear model. *Neuroimage* 92:381–397.

Winkler, A.M., G.R. Ridgway, G. Douau, T.E. Nichols, and S.M. Smith. 2016. Faster permutation inference in brain imaging. *NeuroImage* 141:502–516.

Worsley, K.J., A.C. Evans, S. Marrett, and P. Neelin. 1992. A three-dimensional statistical analysis for cbf activation studies in human brain. *Journal of Cerebral Blood Flow Metabolism* 12:900–918.

Worsley, K.J., S. Marrett, P. Neelin, A.C. Vandal, K.J. Friston, and A.C. Evans. 1996. A unified statistical approach for determining significant signals in images of cerebral activation. *Human Brain Mapping* 4:58–73.

Wright, John, Arvind Ganesh, Shankar Rao, Yigang Peng, and Yi Ma. 2009. Robust principal component analysis: Exact recovery of corrupted low-rank matrices via convex optimization. In *Advances in neural information processing systems*, 2080–2088.

Xu, Jia, Vamsi K Ithapu, Lopamudra Mukherjee, James M Rehg, and Vikas Singh. 2013. Gosus: Grassmannian online subspace updates with structured-sparsity. In *Proceedings of the ieee international conference on computer vision*, 3376–3383.

Y. Hochberg, A.C. Tamhane. 1987. *Multiple comparison procedures*. 1st ed. Wiley.

Yan, Zhicheng, Hao Zhang, Robinson Piramuthu, Vignesh Jagadeesh, Dennis DeCoste, Wei Di, and Yizhou Yu. 2015. Hd-cnn: hierarchical deep convolutional neural networks for large scale visual recognition. In *Proceedings of the ieee international conference on computer vision*, 2740–2748.

Yosinski, Jason, Jeff Clune, Yoshua Bengio, and Hod Lipson. 2014. How transferable are features in deep neural networks? In *Advances in neural information processing systems*, 3320–3328.

Yosinski, Jason, Jeff Clune, Anh Nguyen, Thomas Fuchs, and Hod Lipson. 2015. Understanding neural networks through deep visualization. *arXiv preprint arXiv:1506.06579*.

Yu, Peng, Jia Sun, Robin Wolz, Diane Stephenson, James Brewer, Nick C Fox, Patricia E Cole, Clifford R Jack, Derek LG Hill, Adam J Schwarz, et al. 2014. Operationalizing hippocampal volume as an enrichment biomarker for amnestic mild cognitive impairment trials: effect of algorithm, test-retest variability, and cut point on trial cost, duration, and sample size. *Neurobiology of aging* 35(4):808–818.

Zeiler, Matthew D, and Rob Fergus. 2014. Visualizing and understanding convolutional networks. In *European conference on computer vision*, 818–833. Springer.

Zhang, Daoqiang, Yaping Wang, Luping Zhou, Hong Yuan, Dinggang Shen, Alzheimer's Disease Neuroimaging Initiative, et al. 2011a. Multimodal classification of alzheimer's disease and mild cognitive impairment. *Neuroimage* 55(3):856–867.

Zhang, Daoqiang, Yaping Wang, Luping Zhou, et al. 2011b. Multimodal classification of Alzheimer's disease and mild cognitive impairment. *Neuroimage* 55(3):856–867.

Zhang, Tianzhu, Bernard Ghanem, and Narendra Ahuja. 2012. Robust multi-object tracking via cross-domain contextual information for sports video analysis. In *2012 ieee international conference on acoustics, speech and signal processing (icassp)*, 985–988. IEEE.

Zhang, Yuchen, John Duchi, and Martin Wainwright. 2013. Divide and conquer kernel ridge regression. In *Conference on learning theory*, 592–617.

Zhou, Hao, Vamsi K Ithapu, Sathya Narayanan Ravi, Vikas Singh, Grace Wahba, and Sterling C Johnson. 2016. Hypothesis testing in unsupervised domain adaptation with applications in alzheimer's disease. In *Advances in neural information processing systems*, 2496–2504.

Zou, Hui, Trevor Hastie, and Robert Tibshirani. 2006. Sparse principal component analysis. *Journal of computational and graphical statistics* 15(2):265–286.