LOSS LANDSCAPES AND GENERALIZATION IN NEURAL NETWORKS: THEORY AND APPLICATIONS

by

Akshay Rangamani

A dissertation submitted to The Johns Hopkins University

in conformity with the requirements for the degree of

Doctor of Philosophy

Baltimore, Maryland

December, 2019

Abstract

In the last decade or so, deep learning has revolutionized entire domains of machine learning. Neural networks have helped achieve significant improvements in computer vision, machine translation, speech recognition, etc. These powerful empirical demonstrations leave a wide gap between our current theoretical understanding of neural networks and their practical performance. The theoretical questions in deep learning can be put under three broad but inter-related themes: 1) Architecture/Representation, 2) Optimization, and 3) Generalization. In this dissertation, we study the landscapes of different deep learning problems to answer questions in the above themes.

First, in order to understand what representations can be learned by neural networks, we study simple Autoencoder networks with one hidden layer of rectified linear units. We connect autoencoders to the well-known problem in signal processing of Sparse Coding. We show that the squared reconstruction error loss function has a critical point at the ground truth dictionary under an appropriate generative model.

Next, we turn our attention to a problem at the intersection of optimization and generalization. Training deep networks through empirical risk minimization is a non-convex problem with many local minima in the loss landscape.

A number of empirical studies have observed that "flat minima" for neural networks tend to generalize better than sharper minima. However, quantifying the flatness or sharpness of minima has been an issue due to possible rescaling in neural networks with positively homogenous activations. We use ideas from Riemannian geometry to define a new measure of flatness that is invariant to rescaling. We test the hypothesis that flatter minima generalize better through a number of different experiments on deep networks.

Finally we apply deep networks to computer vision problems with compressed measurements of natural images and videos. We conduct experiments to characterize the situations in which these networks fail, and those in which they succeed. We train deep networks to perform object detection and classification directly on these compressive measurements of images, without trying to reconstruct the scene first. These experiments are conducted on public datasets as well as datasets specific to a sponsor of our research.

Thesis Committee

Readers

Trac D. Tran (Primary Advisor)
Professor
Department of Electrical and Computer Engineering
Johns Hopkins Whiting School of Engineering

Vishal Patel
Assistant Professor
Department of Electrical and Computer Engineering
Johns Hopkins Whiting School of Engineering

Dissertation Defense Committee Member

Rene Vidal
Professor
Department of Biomedical Engineering
Johns Hopkins Whiting School of Engineering

Acknowledgments

Working towards a doctorate degree has been one of the hardest and most rewarding experiences of my life, and I could not have done this without the assistance and encouragement of whole universe of people.

I would first like to thank my advisor Dr. Trac D. Tran for his guidance, mentorship, and support over the six years that I spent in the Digital Signal Processing Lab. He gave me the freedom to explore widely and define the problems that I worked on during my Ph.D. Dr. Tran was generous enough to let me first explore applications of deep learning and later pivot to theory when some of those initiatives were not fruitful. Dr. Tran mentored me in research and also gave me an opportunity to serve as a teaching assistant for his courses on Compressed Sensing and Introduction to Electrical and Computer Engineering. His support for my conference and workshop travel and his motivating my internship experiences were also formative in my graduate career.

I would also like to thank Dr. Sang Peter Chin, who was also a valuable mentor during my time at JHU. Dr. Chin introduced me to intriguing problems in deep learning, and encouraged me to think about how to formulate problems in unfamiliar domains.

Dr. Raman Arora was an essential part of my graduate career, it was through interacting with him inside and outside of courses that I developed an appreciation for and a liking towards statistical learning theory, and machine learning. Taking part in his reading groups helped me keep abreast of the fire hose of new material that comes out in current machine learning research. I also appreciate the opportunities he gave me to be a teaching assistant for the popular machine learning course at JHU. I would like to thank him for serving on my Graduate Board Oral committee as well as my thesis proposal committee.

Dr. Amitabh Basu was another helpful mentor and collaborator during my time at JHU. I really appreciated his course on Convexity, and enjoyed working with him during our work on Sparse Coding and Autoencoders. His mathematical rigor and insightful questions helped me iron out the gaps and kinks in my thinking and present a more rigorous paper.

I would also like to thank Dr. Rene Vidal and Dr. Vishal Patel for serving on my dissertation defense and thesis proposal committees. Their comments and suggestions helped me strengthen this dissertation.

One of my most valuable experiences during my graduate career was the opportunity I got to intern at the T.J Watson Research Center at IBM. I loved my time there, juggling deep learning research together with working on more practical applications. I am indebted to Dr. Nam H. Nguyen for the opportunity and mentorship he gave me during the course of my internship. I am also greatly thankful for the opportunity to interact deeply with Dr. Abhishek Kumar (currently at Google) who is a great collaborator and genial personality. I also learned a lot about developing a large software package and presenting how algorithms work in an interesting manner thanks to Dr. Dhaval Patel, and Dr. Jayant Kalagnanam.

Over my years at JHU I have been fortunate to have interacted with many smart and capable colleagues. I would like to thank Anirbit Mukherjee, with whom I worked on the theme of Chapter 2 of this dissertation, for being a committed collaborator. I would also like to thank Poorya Mianjy and Teodor Marinov for many an intellectually stimulating discussion. I am also grateful to have met Jonathan Jones, Dr. Keith Levin, Dr. Joshua Cape, Dr. Kamel Lahouel, and other members of our information geometry reading group, from which I learned a lot, and which served as a useful detour from my dissertation research. I am grateful to the members of the DSP lab for providing a warm, welcoming, and supportive environment for research, including Dr. Yuanming Suo, Dr. Xiaoxia Sun, Dr. Minh Dao, Dr. Dung Tran, Dr. Tao Xiong, Dr. Xiang Xiang, Dr. Luoluo Liu, Arun Nair, Sonia Joy, and most recently Minh Bui. I would be remiss if I did not mention the wonderful folks at Johns Hopkins ECE, including other co-founders of the ECE Graduate Student Association - Dr. Kayode Sanni, Dr. Kate Fischl, Dr. Ebuka Arinze, Gaspar Tognetti, Michelle Graham, Alycen Wiacek, Niharika Shimona D'Souza, Ranjani Srinivasan, Jeff Craley, Naresh Nandakumar, etc.

I have had the good fortune to live with wonderful roommates and lifelong friends in Dr. Varun Chokshi and Dr. Shourya Sonkar Roy Burman. Times spent hanging out with them and Sravya Kurapati and Rujuta Deshpande were always joyful. A large contingent of friends have made living in Baltimore a lively and enjoyable time - Purnima Padmanabhan, Mithra Kumar, Dr. Veena Padmanabhan, Dr. Ramanujan Srinath, Nagaraj Mahajan, Dr. Vimal Manohar, Vivek Nagal, Niti Agarwal, Dr. Anand Malpani, Princy Parsana, Dr. Siddharth Mahendran, Dr. Vishwa Parekh, Garima Shah, Shruthi Bhat, etc. My tight-knit circle of undergraduate classmates (Dr. Milind Rao, Karthik Rajkumar, Sujan Kumar, Saurabh Sawlani, and Anand Rao) were a great source of support, and other friends from IIT Madras (Arun Vinayak, Sithara Rasheed, Vinay Sridhar, and Padmapriya Govindarajan) were valuable motivators.

Finally, I would like to thank my parents - Rangamani Rajan and Sripriya Rangamani, as well as my brother - Adithya for coming along this journey with me. They have been around to share in my successes and comfort me during the low points of graduate school. I would also like to thank other close members of my family - my grandparents, my aunt Divya, cousin Anjali, uncle Ramaprasad, and cousin Abhijith.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

Machine Learning is a powerful computational paradigm that uses the large amounts of data being collected to make predictions and inferences about the world around us. It is at the core of a data-driven way of understanding the world, that uses datasets to develop models, usually without much innate structure, to understand and predict phenomena occurring in our world.

Machine learning can be broadly defined as an approach to developing computational methods that *perform* better on a *task* with more *experience*. We can make this more precise by formalizing what we mean by each of the italicized terms in the previous definition. In the supervised learning paradigm, for instance, our *task* is to find a mapping $f : \mathcal{X} \to \mathcal{Y}$ between *inputs* $\mathbf{x}$ in an input domain $\mathcal{X}$ and *outputs* $\mathbf{y}$ in an output domain $\mathcal{Y}$. To find a "good" mapping we are given a set of examples $\mathcal{S} = \{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=1}^{N}$ that are drawn from a distribution $\mathcal{D}$. These examples signify the *experience* that our machine learning algorithm gains. The *performance* of our chosen mapping on this task is usually measured by a *loss function* that can score how close our mapping $f(\mathbf{x})$ comes to predicting the desired output $\mathbf{y}$ as $\ell(f(\mathbf{x}), \mathbf{y})$. The goal of machine learning is to find a mapping f that performs well not only on the sample $\mathcal{S}$, but also *generalizes* to the population:

$$f^* = \operatorname{argmin} \mathbb{E}_{(\mathbf{x},\mathbf{y})\sim\mathcal{D}} \left[\ell(f(\mathbf{x}), \mathbf{y})\right]$$

Of course searching for the best mapping over all possible mappings in $\mathcal{Y}^{\mathcal{X}}$ is a daunting task, so we typically restrict our search to a certain class of functions - linear, polynomial, etc. and try to find the best function within that class. Different function classes give us different types of machine learning algorithms, like linear regression, polynomial regression, support vector machines, among others. In most of these formulations we usually have parametric forms of the function classes, which means that solving our machine learning problem boils down to finding the best set of parameters.

$$\mathbf{w}^* = \operatorname*{argmin}_{\mathbf{w}\in\mathcal{W}\subseteq\mathbb{R}^d} \mathbb{E}_{(\mathbf{x},\mathbf{y})\sim\mathcal{D}} \left[\ell(f_{\mathbf{w}}(\mathbf{x}), \mathbf{y})\right]$$

Neural networks are a specific kind of function class that consist of multiple layers of linear transformations followed by nonlinearities. The linear transformations could be straightforward matrix multiplications, or convolutions, etc. The nonlinearities are usually applied entrywise to each layer. Common nonlinearities used in neural networks are $\tanh(x)$, $\text{sigmoid}(x) = (1 + e^{-x})^{-1}$, or $\text{ReLU}(x) = \max(x, 0)$, among others. Deep learning is usually used to refer to the practice of machine learning using deep (many-layered) neural networks along with gradient-based optimization techniques to find the best functions.

$$f_{\mathrm{W}}(\mathbf{x}) = \mathrm{W}_L\phi(\mathrm{W}_{L-1}\phi(\ldots \mathrm{W}_2\phi(\mathrm{W}_1\mathbf{x} + \mathbf{b}_1) + \mathbf{b}_2 \ldots) + \mathbf{b}_{L-1}) + \mathbf{b}_L$$

In the last decade or so, deep learning (LeCun, Bengio, and Hinton, 2015) and neural networks have revolutionized entire domains of machine learning. They have helped achieve significant improvements in computer vision (Krizhevsky, Sutskever, and Hinton, 2012; Ren et al., 2015), machine translation (Sutskever, Vinyals, and Le, 2014; Jean et al., 2014), speech recognition (Sainath et al., 2013; Hinton et al., 2012), etc. These powerful empirical demonstrations leave a wide gap between our current theoretical understanding of neural networks, and their practical performance.

It is useful to divide the theoretical questions in deep learning into three broad themes: Architecture/Representations, Optimization, and Generalization (this framework was popularized by Vidal et al., 2017 among others).

Architecture/Representation: There are a lot of choices to be made while designing neural networks for machine learning problems. The number of layers to use, the size of each layer in the network, the types of nonlinearities to use, and so on. The approximation properties of the networks is influenced by these design choices, as well as the type of transformations that the network applies to the data and the type of representations that different network designs can learn. Understanding the effects of different architectural design choices is thus one key piece of a theory of deep learning.

Optimization: Even though our goal in machine learning is to do well on a population, we only have access to the population through samples in $\mathcal{S}$. A popular framework used to solve machine learning problems is the *empirical risk minimization* framework, in which the mapping (or parameter vector) that minimizes the empirical average of the loss function over the sample $\mathcal{S}$ is chosen

$$\mathbf{w}^{ERM} = \underset{\mathbf{w}\in\mathcal{W}\subseteq\mathbb{R}^d}{\text{argmin}} \frac{1}{N}\sum_{i=1}^{N} \ell(f_{\mathbf{w}}(\mathbf{x}_i), \mathbf{y}_i)$$

Deep learning also involves optimizing the sample average of the loss function (possibly with some regularization added) over the space of parameters. Due to the layered structure of deep networks, the objective function in these spaces is non-convex, making this a hard problem to solve. Understanding how to solve these high-dimensional, non-convex optimization problems to global optimality is thus another component in understanding deep learning.

Generalization: While in most approaches to machine learning we minimize the sample average of our loss function, it is always important to remember that our actual goal is to perform well on the population. We would thus like to be able to characterize how solutions obtained through (regularized) empirical risk minimization will perform on unseen examples that are drawn from the same distribution.

These themes are only a rough delineation of the types of questions that can be asked about understanding deep learning from a theoretical perspective. There are questions that can fall under more than one theme and results from problems in each of these themes have implications beyond just the bucket that they fall under.

1.1 Thesis Contributions and Outline

Our main contributions and the bulk of this dissertation are described below:

1. **Representations:** Neural networks are not only useful in supervised learning, but are also used to learn representations that might be useful in understanding a dataset or useful in a downstream classification/regression task. Autoencoders are a type of neural network architecture in which the networks try to reconstruct the input from nonlinear transformations of the input. They are used to learn representations of data in a wide range of applications for vision, speech, time series analysis, etc. Autoencoders have been used either as building blocks within larger data analysis pipelines, or even for layer-wise pre-training of deeper networks. Understanding what sort of representations they can learn is an important step to consider in applying autoencoders to different problems.

   Through analysis of the loss landscape of autoencoders, we establish connections between autoencoders and *Sparse Coding* or *Dictionary Learning*, a well known problem in signal processing (Rangamani et al., 2018). We show that under a sparse coding generative model, the landscape of the squared reconstruction error of a ReLU autoencoder has a critical point at the ground truth dictionary. Simulations also tell us that if we start a gradient descent algorithm far away from the ground truth dictionary, we end up close to it after enough iterations. This theoretical investigation supported by simulations tells us that when we train ReLU autoencoders, we are essentially solving a sparse coding problem. This work is presented in **Chapter 2**.

2. **Generalization:** One heuristic that tries to explain why deep networks are able to generalize is that training algorithms like stochastic gradient descent tend to drive the parameters of the network to shallow, wide wells that are referred to as "flat minima" (Keskar et al., 2016). While this observation has been around for a long time (Hochreiter and Schmidhuber, 1997), it was recently shown (Dinh et al., 2017) that for deep networks with positively homogenous activation functions (like ReLU) quantitative measures of "flatness" could be made arbitrarily large or small through a simple rescaling of the deep network.

   In order to be able to test whether "flatness" is indeed correlated with generalization, we first need a procedure to measure flatness quantitatively. By approaching this problem using techniques from manifold geometry, we propose a flatness metric (Rangamani et al., 2019) that is invariant to these rescalings. We then apply this technique to compare minima obtained using large-batch and small-batch gradient based methods, and was able to empirically confirm the observation that "flatter minima" generalize better. Our work is one of the first to consider the space of deep network parameters as a differentiable quotient manifold rather than a Euclidean space. This work is presented in **Chapter 3**.

3. **Computer Vision with Compressive Measurements:** In resource-constrained

situations, compressive sensing can help us reduce the amount of data we have to collect and transmit. We build an object detection and tracking system based on deep networks that can work with a custom image sensor Zhang et al., 2016 that collects compressive measurements of scenes. We analyze the scenes directly using the compressive measurements, instead of trying to reconstruct the images and then performing object detection/classification. We propose and empirically atudy some training schemes that allow us to adapt deep networks for object detection from natural images to our setting Nair et al., 2018. We also apply these networks to video sequences in specialized settings that come from our research sponsor. We present this work in **Chapter 4**.

References

LeCun, Yann, Yoshua Bengio, and Geoffrey Hinton (2015). "Deep learning". In: *nature* 521.7553, p. 436.

Krizhevsky, Alex, Ilya Sutskever, and Geoffrey E Hinton (2012). "Imagenet classification with deep convolutional neural networks". In: *Advances in neural information processing systems*, pp. 1097–1105.

Ren, Shaoqing, Kaiming He, Ross Girshick, and Jian Sun (2015). "Faster r-cnn: Towards real-time object detection with region proposal networks". In: *Advances in neural information processing systems*, pp. 91–99.

Sutskever, Ilya, Oriol Vinyals, and Quoc V Le (2014). "Sequence to sequence learning with neural networks". In: *Advances in neural information processing systems*, pp. 3104–3112.

Jean, Sébastien, Kyunghyun Cho, Roland Memisevic, and Yoshua Bengio (2014). "On using very large target vocabulary for neural machine translation". In: *arXiv preprint arXiv:1412.2007*.

Sainath, Tara N, Abdel-rahman Mohamed, Brian Kingsbury, and Bhuvana Ramabhadran (2013). "Deep convolutional neural networks for LVCSR". In: *Acoustics, speech and signal processing (ICASSP), 2013 IEEE international conference on*. IEEE, pp. 8614–8618.

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". In: *IEEE Signal processing magazine* 29.6, pp. 82–97.

Vidal, Rene, Joan Bruna, Raja Giryes, and Stefano Soatto (2017). "Mathematics of deep learning". In: *arXiv preprint arXiv:1712.04741*.

Rangamani, Akshay, Anirbit Mukherjee, Amitabh Basu, Ashish Arora, Tejaswini Ganapathi, Sang Chin, and Trac D Tran (2018). "Sparse coding and autoencoders". In: *2018 IEEE International Symposium on Information Theory (ISIT)*. IEEE, pp. 36–40.

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". In: *arXiv preprint arXiv:1609.04836*.

Hochreiter, Sepp and Jürgen Schmidhuber (1997). "Flat minima". In: *Neural Computation* 9.1, pp. 1–42.

Dinh, Laurent, Razvan Pascanu, Samy Bengio, and Yoshua Bengio (2017). "Sharp minima can generalize for deep nets". In: *arXiv preprint arXiv:1703.04933*.

Rangamani, Akshay, Nam H Nguyen, Abhishek Kumar, Dzung Phan, Sang H Chin, and Trac D Tran (2019). "A Scale Invariant Flatness Measure for Deep Network Minima". In: *arXiv preprint arXiv:1902.02434*.

Zhang, Jie, Tao Xiong, Trac Tran, Sang Chin, and Ralph Etienne-Cummings (2016). "Compact all-CMOS spatiotemporal compressive sensing video camera with pixel-wise coded exposure". In: *Optics express* 24.8, pp. 9013–9024.

Nair, Arun Asokan, Luoluo Liu, Akshay Rangamani, Peter Chin, Muyinatu A. Lediju Bell, and Trac D. Tran (2018). "Reconstruction-free deep convolutional neural networks for partially observed images". In: *2018 IEEE Global Conference on Signal and Information Processing (GlobalSIP)*. IEEE.

Chapter 2

Sparse Coding and Autoencoders

Machine learning problems can broadly be classified as supervised or unsupervised learning problems. In unsupervised learning problems, we usually do not have access to labels for data that we want to learn from. This means our task is also not the same as it is in supervised learning, which is finding a mapping from an input space to a label space. As a consequence there are many different kinds of tasks which can fall under the category of unsupervised learning. One important task is learning *representations* of data that can be useful in downstream tasks, for instance, if we would like to classify the data or perform clustering. *Principal Components Analysis* and *Independent Components Analysis* are familiar instances of representation learning problems.

Neural networks and deep learning are powerful approaches to solve unsupervised learning problems as well. One of the fundamental neural network based approaches for representation learning is the *Autoencoder*, which attempts to reconstruct its input from nonlinear transformations of the same. In the earlier parts of the resurgence of deep learning, Autoencoders were used to pre-train layers of deep neural networks and find better initializations of deep architectures. While autoencoders were ubiquitous objects in machine learning and representation learning, the question of what kinds of representations they could learn was still open. In this chapter we answer this question by showing that expansive autoencoders (those with hidden layers larger than the input dimension) can solve the Sparse Coding problem, and can learn overcomplete, incoherent Dictionaries. We answer this question by analyzing the loss landscape of the objective used to train autoencoders. The work in this chapter was presented earlier in Rangamani et al., 2018

2.1 Introduction

In *Dictionary Learning/Sparse Coding* one receives samples of vectors $y \in \mathbb{R}^n$ that have been generated as $y_i = A^* x_i^*$ where $A^* \in \mathbb{R}^{n \times h}$ and $x_i^* \in \mathbb{R}^h$ and $h > n$. We typically assume that the number of non-zero entries in x_i^* to be no larger than some sub-linear function of the dimension h and that A^* satisfies certain incoherence properties. The question now is to recover A^* from the samples y_i. There have been renewed investigations into the hardness of this problem (Tillmann, 2015) and many former results have recently been reviewed in (Gilbert, 2017; Schnass, 2015a). Ever since the ground-breaking paper, (Olshausen and Field, 1997) (a recent review by the same authors can be found in Olshausen and Field, 2005), many algorithms have been developed to solve sparse coding in both heuristic and provable approaches - Donoho and Huo, 2001; Aharon, Elad, and Bruckstein, 2006; Spielman, Wang, and Wright, 2012; Błasiok and Nelson, 2016; Agarwal et al., 2014; Arora et al., 2015; Sun, Qu, and Wright, 2015; Barak, Kelner, and Steurer, 2014; Remi and Schnass, 2010; Geng and Wright, 2014; Schnass, 2015b. A detailed comparison among these various approaches can be found in Błasiok and Nelson, 2016.

Recent investigations have led to the conjecture/belief that many neural unsupervised learning approaches are sparse coding problems in disguise (Makhzani and Frey, 2013; Makhzani and Frey, 2015). Olshausen and Field had already made the connection between sparse coding and training neural architectures and in today's terminology this is reminiscent of the architecture of an autoencoder (Olshausen and Field, 1996). Provable training of neural nets has been a long standing open question and many recent works have focussed on such proofs for nets with one hidden layer (Li and Yuan, 2017; Tian, 2017) However, to the best of our knowledge it is not clear if the techniques in these papers can be adapted to analyze the landscape of a loss function for autoencoders. In this chapter we bridge the gap between autoencoders and sparse coding. Specifically, we investigate the landscape of the squared loss function of an autoencoder in the vicinity of the ground truth dictionary, and make progress towards understanding whether gradient descent on autoencoder architectures can solve the *Dictionary Learning* problem.

In the rest of this chapter, we will discuss our results. In section 2.2 we define our autoencoder model as well as the generative model for our data. In section 2.3 we present our main results. Section 2.4 presents the proof of 2.3.1 and section 2.5 presents the proof of our main theorem 2.3.2. We verify our theoretical results through simulations in 4.4, and summarize the chapter in section 2.7

2.2 Introducing the neural architecture and the distributional assumptions

For any $n, h \in \{1, 2, ..\}$, we consider autoencoders which are fully connected $\mathbb{R}^n \to \mathbb{R}^n$ neural networks with a single hidden layer of h activations. We focus on networks that use the Rectified Linear Unit (ReLU) activation which is the function ReLU : $\mathbb{R}^h \to \mathbb{R}^h$ mapping $\mathbf{x} \to (\max\{0, x_i\})_{i=1}^h$. In this case, the autoencoder can be seen as computing the following function $\hat{\mathbf{y}}(\mathrm{W}, \mathbf{y}, \epsilon)$ as follows,

$$\mathbf{r} = \mathrm{ReLU}\left(\mathrm{W}\mathbf{y} - \epsilon\right)$$

$$\hat{\mathbf{y}} = \mathrm{W}^\top \mathbf{r} \tag{2.1}$$

Here $\mathbf{y} \in \mathbb{R}^n$ is the input to the autoencoder, $\mathrm{W} \in \mathbb{R}^{h \times n}$ is the linear transformation implemented by the first layer, $\mathbf{r} \in \mathbb{R}^h$ is the output of the layer of activations, $\epsilon \in \mathbb{R}^h$ is the bias vector and $\hat{\mathbf{y}} \in \mathbb{R}^n$ is the output of the autoencoder. Note that we impose the condition that the second layer of weights is simply the transpose of the first layer; this setting with tied weights turned out to yield the desired connections to sparse coding. We define the columns of $\mathrm{W}^\top$ (rows of W) as $\{W_i\}_{i=1}^h$.

2.2.1 Assumptions on the dictionary and the sparse code

We assume that our signal $\mathbf{y}$ is generated using sparse linear combinations of atoms/vectors of an overcomplete dictionary, i.e., $\mathbf{y} = \mathrm{A}^*\mathbf{x}^*$, where $\mathrm{A}^* \in \mathbb{R}^{n \times h}$ is a dictionary, and $\mathbf{x}^*$ is a compactly supported non-negative sparse

code. $\mathbf{x}^*$ is assumed to have at most $k = h^p$ (for some $0 < p < 1$) non-zero elements and each non-zero coordinate of $\mathbf{x}^*$ is contained in $[a(h), b(h)]$ with $a(h) > 0$. The columns of the original dictionary A^* (labeled as $\{A_i^*\}_{i=1}^h$) are assumed to be normalized and also satisfying the incoherence property such that $\max_{\substack{i,j=1,..,h \\ i \neq j}} |\langle A_i^*, A_j^* \rangle| \leq \frac{\mu}{\sqrt{n}} = h^{-\xi}$ for some $\xi > 0$.
We assume that the sparse code $\mathbf{x}^*$ is sampled from a distribution with the following properties. We fix a set of possible supports of $\mathbf{x}^*$, denoted by $\mathbb{S} \subseteq 2^{[h]}$, where each element of $\mathbb{S}$ has at most $k = h^p$ elements. We consider any arbitrary discrete probability distribution $D_{\mathbb{S}}$ on $\mathbb{S}$ such that the probability $q_1 := \mathbb{P}_{S \sim \mathbb{S}}[i \in S]$ is the same for all $i \in [h]$, and the probability $q_2 := \mathbb{P}_{S \in \mathbb{S}}[i, j \in S]$ is the same for all $i, j \in [h]$. A special case is when $\mathbb{S}$ is the set of all subsets of size k, and $D_{\mathbb{S}}$ is the uniform distribution on $\mathbb{S}$. For every $S \in \mathbb{S}$ there is a distribution say D_S on $(\mathbb{R}^{\geq 0})^h$ which is supported on vectors whose support is contained in S and which is uncorrelated for pairs of coordinates $i, j \in S$. Further, we assume that the distributions D_S are such that each coordinate i is compactly supported over an interval $[a(h), b(h)]$, where $a(h)$ and $b(h)$ are independent of both i and S but will be functions of h. Moreover, $m_1(h) := \mathbb{E}_{\mathbf{x}^* \sim D_S}[x_i^*]$, and $m_2(h) := \mathbb{E}_{\mathbf{x}^* \sim D_S}[x_i^{*2}]$ are assumed to be independent of both i and S but allowed to depend on h. For ease of notation henceforth we will keep the h dependence of these variables implicit and refer to them as a, b, m_1 and m_2. All of our results will hold in the special case when a, b, m_1, m_2 are constants (no dependence on h).

2.3 Main Results

2.3.1 Recovery of the support of the sparse code by a ReLU layer

First we prove the following theorem which precisely quantifies the sense in which a layer of ReLU gates is able to recover the support of the sparse code when the weight matrix of the deep net is close to the original dictionary.

Theorem 2.3.1 *(***Recovering the Sparse Code Support at the Hidden Layer***)* *Let each column of $W^\top$ be within a δ-ball of the corresponding column of A^*, where $\delta = O\left(h^{-p-\nu^2}\right)$ for some $\nu > 0$, such that $p + \nu^2 < \xi$. We further assume that $a = \omega\left(bh^{-\nu^2}\right)$. Let the bias of the hidden layer of the autoencoder, as defined in (2.1) be $\epsilon = 2m_1 k\left(\delta + \frac{\mu}{\sqrt{n}}\right)$. Let $\mathbf{r}$ be the vector defined in (2.1). Then $r_i \neq 0$ if $i \in$ supp$(\mathbf{x}^*)$, and $r_i = 0$ if $i \notin$ supp$(\mathbf{x}^*)$ with probability at least $1 - \exp\left(-\frac{2h^p m_1^2}{(b-a)^2}\right)$ (with respect to the distribution on $\mathbf{x}^*$).*

As long as $\frac{h^p m_1^2}{(b-a)^2}$ is large, i.e., an increasing function of h, we can interpret this as saying that the probability of the adverse event is small, and we have successfully achieved support recovery at the hidden layer in the limit of large sparse code dimension.

2.3.2 Asymptotic Criticality of the Autoencoder around A^*

In this work we analyze the following standard squared loss function for the autoencoder,

$$L = \frac{1}{2}||\hat{\mathbf{y}} - \mathbf{y}||^2 \tag{2.2}$$

In the above we continue to use the variables as defined in equation 2.1. If we consider a generative model in which A^* is a square, orthonormal matrix and $\mathbf{x}^*$ is a non-negative vector (not necessarily sparse), it is easily seen that the standard squared reconstruction error loss function for the autoencorder has a global minimum at $\mathrm{W} = \mathrm{A}^{*\top}$. However in our generative model A^* is an incoherent and overcomplete dictionary.

Theorem 2.3.2 (**The Main Theorem**) *Assume that the hypotheses of Theorem 2.3.1 hold, and $p < \min\{\frac{1}{2}, \nu^2\}$ (and hence $\xi > 2p$). Further, assume that the distribution parameters are such that $\exp\left(\frac{h^p m_1^2}{2(b-a)^2}\right)$ is superpolynomial in h (which holds, for example, when m_1, a, b are $O(1)$). Then for $i = 1, \ldots, h$,*

$$\left\| \mathbb{E}\left[\frac{\partial L}{\partial W_i}\right] \right\|_2 \leq o\left(\frac{\max\{m_1^2, m_2\}}{h^{1-p}}\right).$$

Since the sparse code dimension (h) is usually large, this theorem tells us that the norm of the gradient is small around A^*. Thus we have shown that this δ ball around A^* is asymptotically (in h) critical for the squared loss function for this autoencoder.

2.4 A Layer of ReLU Gates can Recover the Support of the Sparse Code (Proof of Theorem 2.3.1)

Most sparse coding algorithms are based on an alternating minimization approach, where one iteratively finds a sparse code based on the current estimate of the dictionary, and then uses the estimated sparse code to update the dictionary. The analogue of the sparse coding step in an autoencoder, is the passing through the hidden layer of activations of a certain affine transformation (W which behaves as the current estimate of the dictionary) of the input vectors. We show that under certain stochastic assumptions, the hidden layer of ReLU gates in an autoencoder recovers with high probability the support of the sparse vector which corresponds to the present input.

Proof 2.4.1 (Proof of Theorem 2.3.1) *From the model assumptions, we know that the dictionary A^* is incoherent, and has unit norm columns. So, $|\langle A_i^*, A_j^* \rangle| \leq \frac{\mu}{\sqrt{n}}$ for all $i \neq j$, and $||A_i^*|| = 1$ for all i. This means that for $i \neq j$,*

$$\begin{aligned} |\langle W_i, A_j^* \rangle| &= |\langle W_i - A_i^*, A_j^* \rangle| + |\langle A_i^*, A_j^* \rangle| \\ &\leq ||W_i - A_i^*||_2 ||A_j^*||_2 + \frac{\mu}{\sqrt{n}} \leq (\delta + \frac{\mu}{\sqrt{n}}) \end{aligned} \tag{2.3}$$

Here the second inequality follows from Cauchy-Schwarz.

Otherwise for $i = j$,

$$\langle W_i, A_i^* \rangle = \langle W_i - A_i^*, A_i^* \rangle + \langle A_i^*, A_i^* \rangle = \langle W_i - A_i^*, A_i^* \rangle + 1,$$

and thus,

$$1 - \delta \le \langle W_i, A_i^* \rangle \le 1 + \delta, \tag{2.4}$$

where we use the fact that $|\langle W_i - A_i^*, A_i^* \rangle| \le \delta$.

Let $\mathbf{y} = A^*\mathbf{x}^*$ *and let* S *be the support of* $\mathbf{x}^*$. *Then we define the input to the ReLU activation* $Q - \epsilon = W\mathbf{y} - \epsilon$ *as*

$$\begin{aligned} Q_i &= \sum_{j \in S} \langle W_i, A_j^* \rangle x_j^* = \langle W_i, A_i^* \rangle x_i^* \mathbb{1}_{i \in S} + \sum_{j \in S \setminus i} \langle W_i, A_j^* \rangle x_j^* \\ &= \langle W_i, A_i^* \rangle x_i^* \mathbb{1}_{i \in S} + Z_i \end{aligned}$$

First we try to get bounds on Q_i *when* $i \in supp(x^*)$. *From our assumptions on the distribution of* x_i^* *we have,* $0 < a \le x_i^* \le b$ *and* $\mathbb{E}[x_i^*] = m_1$ *for all* i *in the support of* x^*. *For* $i \in supp(x^*)$,

$$Q_i = \langle W_i, A_i^* \rangle x_i^* + Z_i$$

$$\implies Q_i \ge (1 - \delta)a + Z_i$$

where we use (2.4). *Using* (2.3), Z_i *has the following bounds:*

$$-bk\left(\delta + \frac{\mu}{\sqrt{n}}\right) \le Z_i \le bk\left(\delta + \frac{\mu}{\sqrt{n}}\right)$$

Plugging in the lower bound for Z_i *and the proposed value for the bias, we get*

$$Q_i - \epsilon \ge (1 - \delta)a - bk\left(\delta + \frac{\mu}{\sqrt{n}}\right) - 2m_1 k\left(\delta + \frac{\mu}{\sqrt{n}}\right)$$

For $Q_i - \epsilon > 0$, we need:

$$a > \frac{(b + 2m_1)\left(\delta + \frac{\mu}{\sqrt{n}}\right) k}{1 - \delta}$$

Now plugging in the values for the various quantities, $\frac{\mu}{\sqrt{n}} = h^{-\xi}$ and $k = h^p$ and $\delta = O\left(h^{-p-\nu^2}\right)$, if we have $a = \omega\left(bh^{-\nu^2}\right)$, then $Q_i - \epsilon > 0$.

Now, for $i \notin supp(x^*)$ we would like to analyze the following probability:

$$Pr[Q_i - \epsilon \geq 0 | i \notin supp(x^*)]$$

We first simplify the quantity $Pr[Q_i - \epsilon \geq 0 | i \notin supp(x^*)]$ as follows

$$Pr[Q_i \geq \epsilon | i \notin supp(x^*)] = Pr[Z_i \geq \epsilon]$$

$$= Pr\left[\sum_{j \in S \setminus i} \langle W_i, A_j^* \rangle x_j^* \geq \epsilon\right]$$

Using the Chernoff's bound, we can obtain

$$\begin{aligned}
Pr[Z_i \geq \epsilon] &\leq \inf_{t\geq 0} e^{-t\epsilon}\mathbb{E}\left[\prod_{j\in S\setminus i}\left[e^{t\langle W_i, A_j^*\rangle x_j^*}\right]\right] \\
&= \inf_{t\geq 0} e^{-t\epsilon}\prod_{j\in S\setminus i}\mathbb{E}\left[e^{t\langle W_i, A_j^*\rangle x_j^*}\right] \\
&\leq \inf_{t\geq 0} e^{-t\epsilon}\mathbb{E}^k\left[e^{t\left(\delta+\frac{\mu}{\sqrt{n}}\right)x_j^*}\right] \\
&\leq \inf_{t\geq 0} e^{-t\epsilon}\left(e^{t\left(\delta+\frac{\mu}{\sqrt{n}}\right)m_1}e^{\frac{t^2\left(\delta+\frac{\mu}{\sqrt{n}}\right)^2(b-a)^2}{8}}\right)^k
\end{aligned}$$

where the second inequality follows from (2.3) *and the fact that t and x_i^* are both nonnegative, and the third inequality follows from Hoeffding's Lemma. Next, we also have*

$$\begin{aligned}
Pr[Z_i \geq \epsilon] &\leq \inf_{t\geq 0} e^{-t\left(\epsilon-k\left(\delta+\frac{\mu}{\sqrt{n}}\right)m_1\right)+t^2\frac{k}{8}\left(\delta+\frac{\mu}{\sqrt{n}}\right)^2(b-a)^2} \\
&= e^{-\frac{(\epsilon-k(\delta+\frac{\mu}{\sqrt{n}})m_1)^2}{\frac{k}{2}(\delta+\frac{\mu}{\sqrt{n}})^2(b-a)^2}}.
\end{aligned}$$

Finally, since $k = h^p$ and $\epsilon = 2m_1k\left(\delta+\frac{\mu}{\sqrt{n}}\right)$, we have

$$\exp\left(-\frac{2(\epsilon-km_1(\delta+\frac{\mu}{\sqrt{n}}))^2}{h^p(\delta+\frac{\mu}{\sqrt{n}})^2(b-a)^2}\right) = \exp\left(-\frac{2h^pm_1^2}{(b-a)^2}\right)$$

2.5 Criticality of a neighborhood of A* (Proof of Theorem 2.3.2)

First, we can evaluate the gradient of the squared loss function, with respect to the i-th column of $W^\top$ as:

$$\frac{\partial L}{\partial W_i} = \mathrm{Th}(W_i^\top \mathbf{y} - \epsilon_i) \left[(W_i^\top \mathbf{y} - \epsilon_i) I + \mathbf{y} W_i^\top \right]$$
$$\times \left(\sum_{j=1}^{h} \mathrm{ReLU}(W_j^\top \mathbf{y} - \epsilon_j) W_j - D\mathbf{y} \right)$$

where $\mathrm{Th}(z) = \mathbf{1}_{z>0}$.

It turns out that the expectation of the full gradient of the loss function (2.2) is difficult to analyze directly. Hence corresponding to the true gradient with respect to the $i^{\text{th}}-$column of $\mathrm{W}^\top$ we create a proxy, denoted by $\widehat{\nabla_i L}$, by replacing in the expression for the true expectation $\nabla_i L = \mathbb{E}\left[\frac{\partial L}{\partial W_i}\right]$ every occurrence of the random variable $\mathrm{Th}(W_i^\top y - \epsilon_i) = \mathrm{Th}(W_i^\top A^* x^* - \epsilon_i)$ by the indicator random variable $\mathbf{1}_{i \in \mathrm{supp}(x^*)}$. This proxy is shown to be a good approximant of the expected gradient in the following lemma.

Lemma 2.5.1 *Assume that the hypotheses of Theorem 2.3.1 hold and additionally let b be bounded by a polynomial in h. Then we have for each i (indexing the columns of* $W^\top$*),*

$$\left\| \widehat{\nabla_i L} - \mathbb{E}\left[\frac{\partial L}{\partial W_i}\right] \right\|_2 \leq poly(h) exp\left(-\frac{h^p m_1^2}{2(b-a)^2} \right)$$

Lemma 2.5.2 *Assume that the hypotheses of Theorem 2.3.1 hold, and $p < \min\{\frac{1}{2}, \nu^2\}$ (and hence $\xi > 2p$). Then for each i indexing the columns of $W^\top$, there exist real valued functions α_i and β_i, and a vector e_i such that $\widehat{\nabla_i L} = \alpha_i W_i - \beta_i A_i^* + e_i$, and*

$$\alpha_i = \Theta(m_2 h^{p-1}) + o(m_1^2 h^{p-1})$$

$$\beta_i = \Theta(m_2 h^{p-1}) + o(m_1^2 h^{p-1})$$

$$\alpha_i - \beta_i = o(\max\{m_1^2, m_2\} h^{p-1})$$

$$||e_i||_2 = o(\max\{m_1^2, m_2\} h^{p-1})$$

The proof of lemma 2.5.1 is given in section 2.8 and that of lemma 2.5.2 in 2.9 Given the above results, we are now in a position to assemble the proof of Theorem 2.3.2.

Proof 2.5.3 (Proof of Theorem 2.3.2) *Consider any i indexing the columns of $W^\top$. Recall the definition of the proxy gradient $\widehat{\nabla_i L}$ at the beginning of this section. Let us define $\gamma_i = \widehat{\nabla_i L} - \mathbb{E}\left[\frac{\partial L}{\partial W_i}\right]$. Using α_i, β_i and e_i as defined in Lemma* **??**, *we can write the expectation of the true gradient as, $\mathbb{E}\left[\frac{\partial L}{\partial W_i}\right] = \alpha_i W_i - \beta_i A_i^* + e_i - \gamma_i$. Further, by Lemma* **??**,

$$\|\gamma_i\| \leq poly(h) exp\left(-\frac{h^p m_1^2}{2(b-a)^2}\right).$$

Since $exp\left(\frac{h^p m_1^2}{2(b-a)^2}\right)$ is superpolynomial in h, we obtain

$$
\begin{aligned}
\left\| \mathbb{E}\left[\frac{\partial L}{\partial W_i}\right] \right\|_2 &= ||\alpha_i W_i - \beta_i A_i^* + e_i - \gamma_i||_2 \\
&= ||\alpha_i (W_i - A_i^*) + (\alpha_i - \beta_i) A_i^* + e_i - \gamma_i||_2 \\
&\leq |\alpha_i| \|W_i - A_i^*\|_2 + |\alpha_i - \beta_i| + ||e_i - \gamma_i||_2 \\
&\leq \frac{\Theta(m_2 h^{p-1})}{h^{2p+\theta^2}} + o(\max\{m_1^2, m_2\} h^{p-1}) \\
&+ o(\max\{m_1^2, m_2\} h^{p-1}) \\
&= o(\max\{m_1^2, m_2\} h^{p-1})
\end{aligned}
$$

2.6 Simulations

We conduct some experiments on synthetic data in order to check whether the gradient norm is indeed small within the columnwise δ-ball of A*. We also make some observations about the landscape of the squared loss function, which has implications for being able to recover the ground-truth dictionary A*.

2.6.1 Data Generation Model

We generate random gaussian dictionaries (A*) of size $n \times h$ where $n = 50$, and $h = 256, 512, 1024, 2048$ and 4096. For each h, we generate a dataset containing $N = 5000$ sparse vectors with h^p non-zero entries, for various $p \in [0.01, 0.5]$. In our experiments, the coherence parameter ξ was approximately 0.1. The support of each sparse vector $\mathbf{x}^*$ is drawn uniformly from all sets of indices of size h^p, and the non-zero entries in the sparse vectors are drawn from a uniform distribution between $a = 1$ and $b = 10$. Once we have generated the sparse vectors, we collect them in a matrix $\mathrm{X}^* \in \mathbb{R}^{h \times N}$ and then compute the signals $\mathrm{Y} = \mathrm{A}^*\mathrm{X}^*$. We set up the autoencoder as defined through equation 2.1. We analyze the squared loss function in (2.2) and its gradient with respect to a column of W through their empirical averages over the signals in Y.

2.6.2 Results

Once we have generated the data, we compute the empirical average of the gradient of the loss function in (2.2) at 200 random points which are columnwise $\frac{\delta}{2} = \frac{1}{2h^{2p}}$ away from A^*. We average the gradient over the 200 points which are all at the same distance from A^*, and compare the average column norm of the gradient to h^{p-1}. Our experimental results shown in Table 2.1 demonstrate that the average column norm of the gradient is of the order of h^{p-1} (and thus falling with h for any fixed p) as expected from Theorem 2.3.2.

We also plot the squared loss of the autoencoder along a randomly chosen direction to understand the geometry of the landscape of the loss function around A^*. We draw a matrix $\Delta\mathrm{W}$ from a standard normal distribution, and normalize its columns. We then plot $f(t) = L(\mathrm{A}^* + t\Delta\mathrm{W}^\top)$, as well as the gradient norm averaged over all the columns. For purposes of illustration, we show these plots for $p = 0.01, 0.1, 0.3$. The plots for $h = 256$ are in Figure 2.1, and those for $h = 4096$ in Figure 2.2. From the plots for $p = 0.01$ and

h \ p	0.01	0.02	0.05	0.1
256	(0.0137, 0.0041)	(0.0138, 0.0044)	(0.0126, 0.0052)	(0.0095, 0.0068)
512	(0.0058, 0.0021)	(0.0058, 0.0022)	(0.0054, 0.0027)	(0.0071, 0.0036)
1024	(0.0025, 0.0010)	(0.0024, 0.0011)	(0.0026, 0.0014)	(0.0079, 0.0020)
2048	(0.0011, 0.0005)	(0.0012, 0.0006)	(0.0025, 0.0007)	(0.0031, 0.0010)
4096	(0.0006, 0.0003)	(0.0012, 0.0003)	(0.0013, 0.0004)	(0.0026, 0.0006)

h \ p	0.2	0.3	0.5
256	(0.0284, 0.0118)	(0.0464, 0.0206)	(0.0343, 0.0625)
512	(0.0104, 0.0068)	(0.0214, 0.0127)	(0.0028, 0.0442)
1024	(0.0078, 0.0039)	(0.0099, 0.0078)	(0.00, 0.0313)
2048	(0.0032, 0.0022)	(0.0036, 0.0048)	(0.00, 0.0221)
4096	(0.0020, 0.0013)	(0.0008, 0.0030)	(0.00, 0.0156)

Table 2.1: Average gradient norm for points that are columnwise $\frac{\delta}{2}$ away from A^*. For each h and p we report $\left(||\mathbb{E}\left[\frac{\partial L}{\partial W_i}\right]||, h^{p-1}\right)$. We note that the gradient norm and h^{p-1} are of the same order, and for any fixed p the gradient norm is decreasing with h as expected from Theorem 2.3.2

0.1, we can observe that the loss function value, and the gradient norm keeps decreasing as we get close to A^*. Figure 2.1 and 2.2 are representative of the shapes obtained for every direction, $\Delta\mathrm{W}$ that we checked. This suggests that A^* might conveniently lie at the bottom of a well in the landscape of the loss function. For the value of $p = 0.3$, (which is much larger than the coherence parameter ξ), Theorem 2.3.1 is no longer valid. We see that the value of the loss function decreases a little as we move away from A^*, and then increases. We suspect that A^* is now in a region where $\mathrm{ReLU}(\mathrm{A}^{*\top}\mathbf{y} - \epsilon) = 0$, which means the function is flat in a small neighborhood of A^*.

We also tried to minimize the squared loss of the autoencoder using gradient descent. In these experiments, we initialized $\mathrm{W}^\top$ far away from A^* (precisely

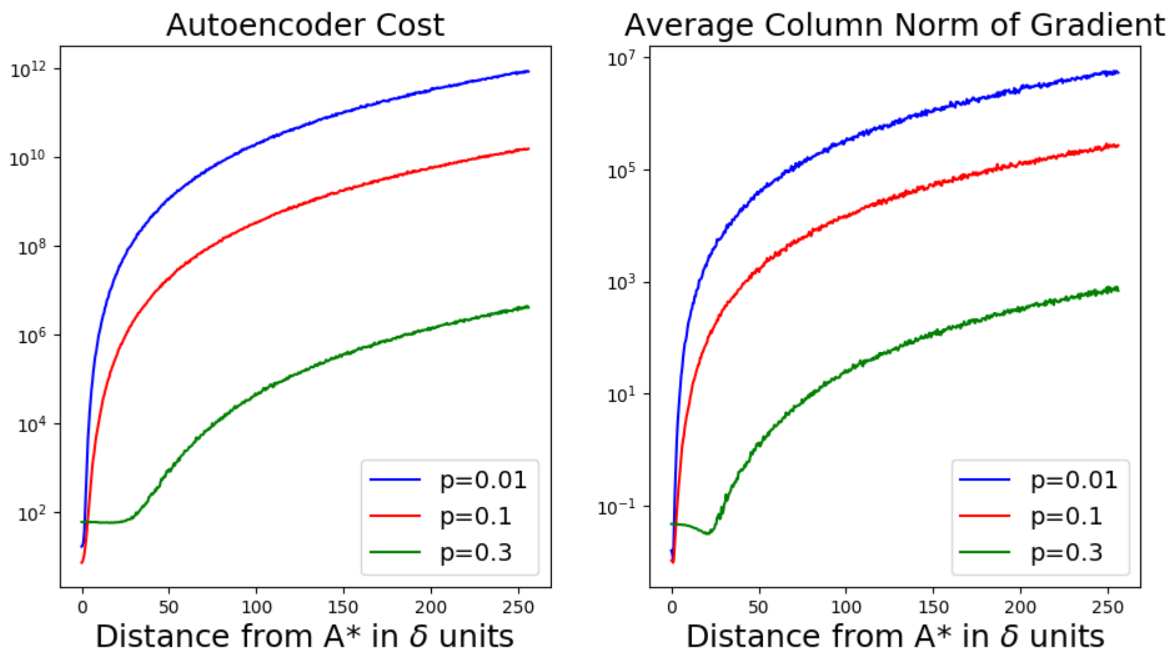


Figure 2.1: Loss function plot for $h = 256, n = 50$

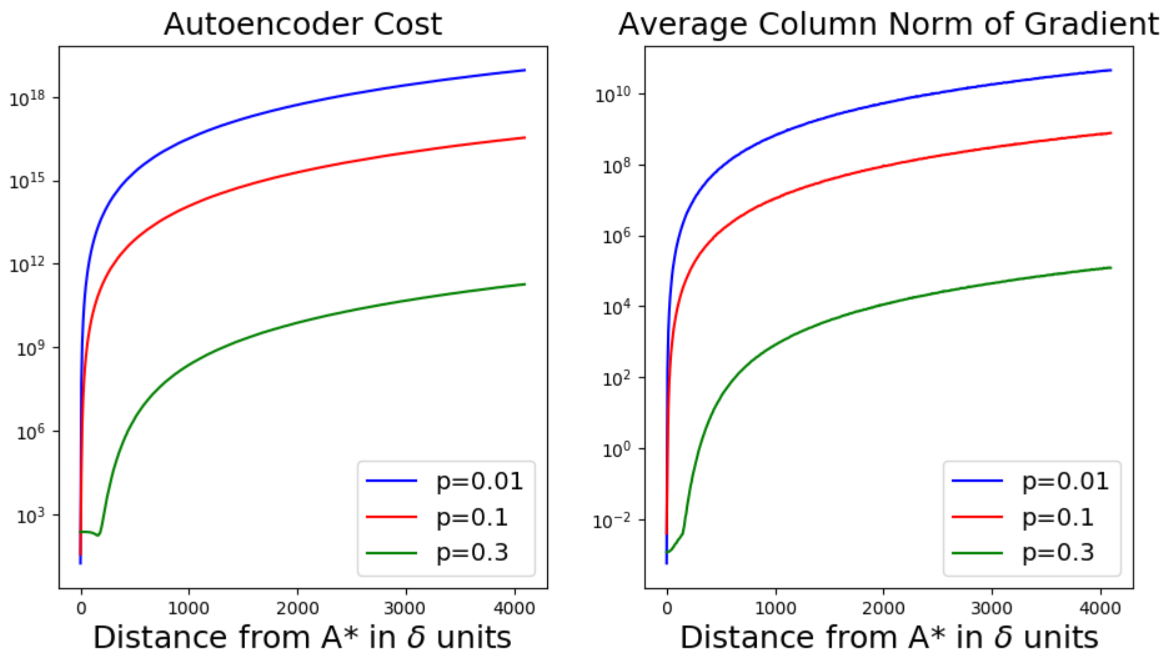


Figure 2.2: Loss function plot for $h = 4096, n = 50$

at a columnwise distance of $\frac{h}{5} \times \delta$), and did gradient descent until the gradient norm dropped below a factor of 2×10^{-5} of the initial norm of the gradient. We then computed the average columnwise distance between $W_{\text{final}}^{\top}$ and A^*, and report the % decrease in the average columnwise distance from the initial point. These results are reported in Table 2.2 below. These experiments suggest that there is a neighborhood of A^* (the radius of which is increasing with h), such that gradient descent initialized at the edge of that neighborhood, greatly reduces the average columnwise distance between $W^{\top}$ and A^*.

h	$p = 0.05$	$p = 0.1$
256	97.7%	96.9%
512	98.6%	98.2%
1024	99%	98.8%
2048	99.2%	99%
4096	99.4%	99.2%

Table 2.2: Fraction of initial columnwise distance covered by the gradient descent procedure

2.7 Conclusion

In this chapter we have undertaken a rigorous analysis of the squared loss of an autoencoder when the data is assumed to be generated by sensing of sparse high dimensional vectors by an overcomplete dictionary. We have proven and have given supporting experiments that the expected gradient of this loss function is very close to zero in a neighborhood of the generating overcomplete dictionary. Our results could shed some light on the observation that gradient descent based algorithms train autoencoders to low reconstruction error for natural data sets, like MNIST.

2.8 The proxy gradient is a good approximation of the true expectation of the gradient (Proof of Lemma 2.5.1)

Proof 2.8.1 *To make it easy to present this argument let us abstractly think of the function f (defined for any $i \in \{1, 2, 3, .., h\}$) as $f(\mathbf{y}, W, X) = \frac{\partial L}{\partial W_i}$ where we have defined the random variable $X = Th[W_i^\top \mathbf{y} - \epsilon_i]$. It is to be noted that because of the ReLU term and its derivative this function f has a dependency on $\mathbf{y} = A^*\mathbf{x}^*$ even outside its dependency through X. Let us define another random variable $Y = \mathbf{1}_{i \in supp(\mathbf{x}^*)}$. Then we have,*

$$
\begin{aligned}
&\left\| \mathbb{E}_{\mathbf{x}^*}[f(\mathbf{y}, W, X)] - \mathbb{E}_{\mathbf{x}^*}[f(\mathbf{y}, W, Y)] \right\|_{\ell_2} \\
\leq &\mathbb{E}_{\mathbf{x}^*}[|f(\mathbf{y}, W, X) - f(\mathbf{y}, W, Y)|_{\ell_2}] \\
\leq &\mathbb{E}_{\mathbf{x}^*}[|f(\mathbf{y}, W, X)(\mathbf{1}_{X=Y} + \mathbf{1}_{X \neq Y}) - f(\mathbf{y}, W, Y)(\mathbf{1}_{X=Y} + \mathbf{1}_{X \neq Y})|_{\ell_2}] \\
\leq &\mathbb{E}_{\mathbf{x}^*}[|(f(\mathbf{y}, W, X) - f(\mathbf{y}, W, Y))|_{\ell_2} \mathbf{1}_{X \neq Y}] \\
\leq &\sqrt{\mathbb{E}_{\mathbf{x}^*}[|f(\mathbf{y}, W, X) - f(\mathbf{y}, W, Y)|_2^2]} \sqrt{\mathbb{E}_{\mathbf{x}^*}[\mathbf{1}_{X \neq Y}]}
\end{aligned}
$$

In the last step above we have used the Cauchy-Schwarz inequality for random variables. We recognize that $\mathbb{E}_{\mathbf{x}^*}[f(\mathbf{y}, W, Y)]$ is precisely what we defined as the proxy gradient $\widehat{\nabla_i L}$. Further for such W as in this lemma the support recovery theorem (Theorem 2.3.1) holds and that is precisely the statement that the term,

$\mathbb{E}_{\mathbf{x}^*}[\mathbf{1}_{X \neq Y}]$ *is small. So we can rewrite the above inequality as,*

$$\left\| \mathbb{E}_{\mathbf{x}^*}[\frac{\partial L}{\partial W_i}] - \widehat{\nabla_i L} \right\|_2 \leq \sqrt{\mathbb{E}_{\mathbf{x}^*}[|f(\mathbf{y}, W, X) - f(\mathbf{y}, W, Y)|_2^2]} \exp\left(-\frac{h^p m_1^2}{2(b-a)^2}\right)$$

We remember that f is a polynomial in h because its h dependency is through Frobenius norms of submatrices of W and ℓ_2 norms of projections of $W\mathbf{y}$. But the ℓ_∞ norm of the training vectors $\mathbf{y}$ (that is b) have been assumed to be bounded by poly(h). Also we have the assumption that the columns of $W^\top$ are within a $\frac{1}{h^{p+\nu^2}}$ −ball of the corresponding columns of A^* which in turn is a $n \times h$ dimensional matrix of bounded norm because all its columns are normalized. So summarizing we have,

$$\left\| \mathbb{E}_{\mathbf{x}^*}[\frac{\partial L}{\partial W_i}] - \widehat{\nabla_i L} \right\|_2 \leq poly(h) \exp\left(-\frac{h^p m_1^2}{2(b-a)^2}\right)$$

The above inequality immediately implies the claimed lemma.

2.9 The asymptotics of the coefficients of the gradient of the squared loss (Proof of Lemma 2.5.2)

To recap we imagine being given as input signals $\mathbf{y} \in \mathbb{R}^n$ (imagined as column vectors), which are generated from an overcomplete dictionary $\mathrm{A}^* \in \mathbb{R}^{n \times h}$ of a fixed incoherence. Let $\mathbf{x}^* \in \mathbb{R}^h$ (imagined as column vectors) be the sparse code that generates $\mathbf{y}$. The model of the autoencoder that we now have is $\hat{\mathbf{y}} = \mathrm{W}^\top \mathrm{ReLU}(\mathrm{W}\mathbf{y} - \epsilon)$. W is a $h \times n$ matrix and the i^{th} column of $\mathrm{W}^\top$ is to be denoted as the column vector W_i.

2.9.1 Derivative of the standard squared loss of a ReLU autoencoder

Using the above notation the squared loss of the autoencoder is $\frac{1}{2}||\hat{\mathbf{y}} - \mathbf{y}||^2$. But we introduce a dummy constant $D = 1$ to be multiplied to $\mathbf{y}$ because this helps read the complicated equations that would now follow. This marker helps easily spot those terms which depend on the sensing of $\mathbf{x}^*$ (those with a factor of D) as opposed to the terms which are "purely" dependent on the neural net (those without the factor of D). Thus we think of the squared loss L of our autoencoder as,

$$L = \frac{1}{2}||\hat{\mathbf{y}} - D\mathbf{y}||^2 = \frac{1}{2}(\mathrm{W}^\top \mathrm{ReLU}(\mathrm{W}\mathbf{y} - \epsilon) - D\mathbf{y})^\top (\mathrm{W}^\top \mathrm{ReLU}(\mathrm{W}\mathbf{y} - \epsilon) - D\mathbf{y}) = \frac{1}{2} f^T f$$

where we have defined $f \in R^n$ as,

$$f = \mathrm{W}^\top \mathrm{ReLU}(\mathrm{W}\mathbf{y} - \epsilon) - D\mathbf{y}$$

Then we have,

$$J_{W_i}(f)_{ab} = \frac{\partial f_a}{\partial W_{ib}} = \text{ReLU}(W_i^\top \mathbf{y} - \epsilon)\delta_{ab} + \text{Th}(W_i^\top \mathbf{y} - \epsilon) W_{ia} y_b$$

In the form of a $n \times n$ derivative matrix this means,

$$J_{W_i}(f) = \left[\frac{\partial f_a}{\partial W_{ib}}\right] = \text{ReLU}(W_i^\top \mathbf{y} - \epsilon) I + \text{Th}(W_i^\top \mathbf{y} - \epsilon) W_i \mathbf{y}^\top$$

This helps us write,

$$\begin{aligned}
\frac{\partial L}{\partial W_i} &= J_{W_i}(f))^\top f \\
&= (\text{ReLU}(W_i^\top \mathbf{y} - \epsilon) I + \text{Th}(W_i^\top \mathbf{y} - \epsilon) W_i \mathbf{y}^\top)^\top [\mathrm{W}^\top \text{ReLU}(\mathrm{W}\mathbf{y} - \epsilon) - D\mathbf{y}] \\
&= \text{Th}(W_i^\top \mathbf{y} - \epsilon_i) \left[(W_i^\top \mathbf{y} - \epsilon_i) I + \mathbf{y} W_i^\top\right] \left(\sum_{j=1}^{h} \text{ReLU}(W_j^\top \mathbf{y} - \epsilon_j) W_j - D\mathbf{y}\right)
\end{aligned}$$

Now going over to the proxy gradient $\widehat{\nabla_i L}$ corresponding to this term we define the vector G_i as,

$$\begin{aligned}
\widehat{\nabla_i L} &= \mathbb{E}_{S \in \mathbb{S}} \left[\mathbf{1}_{i \in S} \times \mathbb{E}_{\mathbf{x}_S^*} \left[\left[(W_i^\top \mathbf{y} - \epsilon_i)\mathrm{I} + \mathbf{y} W_i^\top\right] \left(\sum_{j \in S} (W_j^\top \mathbf{y} - \epsilon_j) W_j - D\mathbf{y}\right)\right]\right] \\
&= \mathbb{E}_{S \in \mathbb{S}} \left[\mathbf{1}_{i \in S} \times G_i\right]
\end{aligned}$$

Thus we have,

$$G_i = \mathbb{E}_{x_S^*}\left[\left[(W_i^\top \mathrm{A}^*\mathbf{x}^* - \epsilon_i)\mathrm{I} + (\mathrm{A}^*\mathbf{x}^*)W_i^\top\right]\left(\sum_{j\in S}(W_j^\top \mathrm{A}^*\mathbf{x}^* - \epsilon_j)W_j - D\mathrm{A}^*\mathbf{x}^*\right)\right]$$

$$= \underbrace{\mathbb{E}_{\mathbf{x}_S^*}\left[(W_i^\top \mathrm{A}^*\mathbf{x}^* - \epsilon_i)\left(\sum_{j\in S}(W_j^\top \mathrm{A}^*\mathbf{x}^* - \epsilon_j)W_j - D\mathrm{A}^*\mathbf{x}^*\right)\right]}_{\text{Term 1}}$$

$$+ \underbrace{\mathbb{E}_{\mathbf{x}_S^*}\left[(\mathrm{A}^*\mathbf{x}^*)W_i^\top\left(\sum_{j\in S}(W_j^\top \mathrm{A}^*\mathbf{x}^* - \epsilon_j)W_j - D\mathrm{A}^*\mathbf{x}^*\right)\right]}_{\text{Term 2}}$$

$$= \underbrace{\mathbb{E}_{\mathbf{x}_S^*}\Bigg[\sum_{j\in S}\epsilon_i\epsilon_j W_j - \sum_{j,k\in S}\epsilon_i(W_j^\top A_k^*)W_j x_k^* - \sum_{j,k\in S}\epsilon_j(W_i^\top A_k^*)W_j x_k^*}_{\text{From Term 1}}$$

$$+ \underbrace{\sum_{j,k,l\in S}(W_i^\top A_k^*)(W_j^\top A_l^*)W_j x_l^* x_k^*\Bigg]}_{\text{From Term 1}}$$

$$+ \underbrace{\mathbb{E}_{x_S^*}\left[-D\sum_{j,k\in S}(W_i^\top A_k^*)A_j^* x_k^* x_j^* + D\sum_{j\in S}\epsilon_i A_j^* x_j^*\right]}_{\text{From Term 1}} + \underbrace{\mathbb{E}_{x_S^*}\left[-D\sum_{j,k\in S}(A_k^{*\top}W_i)A_j^* x_k^* x_j^*\right]}_{\text{From Term 2}}$$

$$+ \underbrace{\mathbb{E}_{x_S^*}\left[-\sum_{j,k\in S}\epsilon_j A_k^*(W_i^\top W_j)x_k^*\right]}_{\text{From Term 2}} + \underbrace{\mathbb{E}_{x_S^*}\left[\sum_{j,k,l\in S}(W_i^\top W_j)(W_j^\top A_l^*)A_k^* x_k^* x_l^*\right]}_{\text{From Term 2}}$$

Now we invoke the distributional assumption about i.i.d sampling of the coordinates for a fixed support and the definition of m_1 and m_2 to write, $\mathbb{E}_{x^*_S}[x^*_i x^*_j] = \mathbb{E}^2_{x^*_S}[x^*_i] = m_1^2$ for all $i \neq j$ and for $i = j$, $m_2 = \mathbb{E}_{x^*_S}[x^*_i x^*_j]$. Thus we get,

$$G_i = \underbrace{\sum_{j \in S} \epsilon_i \epsilon_j W_j - m_1 \sum_{j,k \in S} (W_j^\top A^*_k) W_j \epsilon_i - m_1 \sum_{j,k \in S} \epsilon_j (W_i^\top A^*_k) W_j}_{G_i^1 \text{ From Term 1}}$$

$$+ \underbrace{m_2 \sum_{j,k \in S} (W_i^\top A^*_k)(W_j^\top A^*_k) W_j + m_1^2 \sum_{\substack{j,k,l \in S \\ k \neq l}} (W_i^\top A^*_k)(W_j^\top A^*_l) W_j}_{G_i^2 \text{ From Term 1}}$$

$$+ \underbrace{\left[-D m_1^2 \sum_{\substack{j,k \in S \\ j \neq k}} (W_i^\top A^*_k) A^*_j - D m_2 \sum_{j \in S} (W_i^\top A^*_j) A^*_j + m_1 D \sum_{j \in S} \epsilon_i A^*_j \right]}_{G_i^3 \text{ From Term 1}}$$

$$- \underbrace{\left[D m_1^2 \sum_{\substack{j,k \in S \\ j \neq k}} (A_k^{*\top} W_i) A^*_j + D m_2 \sum_{j \in S} (A_j^{*\top} W_i) A^*_j \right]}_{G_i^4 \text{ From Term 2}}$$

$$\underbrace{- m_1 \left[\sum_{j,k \in S} \epsilon_j (W_i^\top W_j) A^*_k \right] + \left[m_2 \sum_{j,k \in S} (W_i^\top W_j)(W_j^\top A^*_k) A^*_k + m_1^2 \sum_{\substack{j,k,l \in S \\ k \neq l}} (W_i^\top W_j)(W_j^\top A^*_l) A^*_k \right]}_{G_i^5 \text{ From Term 2}}$$

Each term in the above sum is a vector. Now we separate out from the sums the terms which are in the directions of W_i or A_i^* and the rest. We remember that this is being under the condition that $i \in S$. To make this easy to read we do this separation for each line of the above equation separately in a different equation block. Also inside every block we do the separation for each summation term in a separate line.

$$G_i^1 = \sum_{j \in S} \epsilon_i \epsilon_j W_j - m_1 \sum_{j,k \in S} (W_j^\top A_k^*) W_j \epsilon_i - m_1 \sum_{j,k \in S} \epsilon_j (W_i^\top A_k^*) W_j$$

$$= \left[\epsilon_i^2 W_i + \sum_{\substack{j \in S \\ j \neq i}} \epsilon_i \epsilon_j W_j \right]$$

$$- m_1 \left[\sum_{k \in S} \epsilon_i (W_i^\top A_k^*) W_i + \sum_{\substack{j,k \in S \\ j \neq i}} (W_j^\top A_k^*) W_j \epsilon_i \right]$$

$$- m_1 \left[\sum_{k \in S} \epsilon_i (W_i^\top A_k^*) W_i + \sum_{\substack{j,k \in S \\ j \neq i}} \epsilon_j (W_i^\top A_k^*) W_j \right]$$

$$G_i^2 = m_2 \sum_{j,k \in S} (W_i^\top A_k^*)(W_j^\top A_k^*) W_j + m_1^2 \sum_{\substack{j,k,l \in S \\ k \neq l}} (W_i^\top A_k^*)(W_j^\top A_l^*) W_j$$

$$= m_2 \left[\sum_{k \in S} (W_i^\top A_k^*)(W_i^\top A_k^*) W_i + \sum_{\substack{j,k \in S \\ j \neq i}} (W_i^\top A_k^*)(W_j^\top A_k^*) W_j \right]$$

$$+ m_1^2 \left[\sum_{\substack{k,l \in S \\ k \neq l}} (W_i^\top A_k^*)(W_i^\top A_l^*) W_i + \sum_{\substack{j,k,l \in S \\ j \neq i \\ k \neq l}} (W_i^\top A_k^*)(W_j^\top A_l^*) W_j \right]$$

$$G_i^4 = -\left[Dm_1^2 \sum_{\substack{j,k \in S \\ j \neq k}} (A_k^{*\top} W_i) A_j^* + Dm_2 \sum_{j \in S} (A_j^{*\top} W_i) A_j^* \right]$$

$$= -D \left[m_1^2 \sum_{\substack{k \in S \\ k \neq i}} (A_k^{*\top} W_i) A_i^* + m_1^2 \sum_{\substack{j,k \in S \\ j \neq k \\ j \neq i}} (A_k^{*\top} W_i) A_j^* \right]$$

$$- D \left[m_2 (A_i^{*\top} W_i) A_i^* + m_2 \sum_{\substack{j \in S \\ j \neq i}} (A_j^{*\top} W_i) A_j^* \right]$$

$$G_i^5 = -m_1 \left[\sum_{j,k \in S} \epsilon_j (W_i^\top W_j) A_k^* \right]$$

$$+ \left[m_2 \sum_{j,k \in S} (W_i^\top W_j)(W_j^\top A_k^*) A_k^* + m_1^2 \sum_{\substack{j,k,l \in S \\ k \neq l}} (W_i^\top W_j)(W_j^\top A_l^*) A_k^* \right]$$

$$= -m_1 \sum_{j \in S} \epsilon_j (W_i^\top W_j) A_i^* - m_1 \sum_{\substack{j,k \in S \\ k \neq i}} \epsilon_j (W_i^\top W_j) A_k^*$$

$$+ m_2 \sum_{j \in S} (W_i^\top W_j)(W_j^\top A_i^*) A_i^* + m_2 \sum_{\substack{j,k \in S \\ k \neq i}} (W_i^\top W_j)(W_j^\top A_k^*) A_k^*$$

$$+ m_1^2 \sum_{\substack{j,l \in S \\ l \neq i}} (W_i^\top W_j)(W_j^\top A_l^*) A_i^* + m_1^2 \sum_{\substack{j,k,l \in S \\ k \neq i,l}} (W_i^\top W_j)(W_j^\top A_l^*) A_k^*$$

So combining the above we have,

$$\widehat{\nabla_i L} = \alpha_i W_i - \beta_i A_i^* + e_i$$

where,

$$\alpha_i = \mathbb{E}_{S\in \mathbb{S}}\Bigg[\mathbf{1}_{i\in S} \times \Bigg\{ m_2 \sum_{k\in S} (W_i^\top A_k^*)(W_i^\top A_k^*) + m_1^2 \sum_{\substack{k,l\in S\\ k\neq l}} (W_i^\top A_k^*)(W_i^\top A_l^*)$$

$$- 2m_1 \sum_{k\in S} \epsilon_i (W_i^\top A_k^*) + \epsilon_i^2 \Bigg\}\Bigg]$$

$$\beta_i = \mathbb{E}_{S\in \mathbb{S}}\Bigg[\mathbf{1}_{i\in S} \times \Bigg\{ 2Dm_1^2 \sum_{\substack{k\in S\\ k\neq i}} (W_i^\top A_k^*) + 2Dm_2 (W_i^\top A_i^*) - Dm_1 \epsilon_i + m_1 \sum_{j\in S} \epsilon_j (W_i^\top W_j)$$

$$- m_2 \sum_{j\in S} (W_i^\top W_j)(W_j^\top A_i^*) - m_1^2 \sum_{\substack{j,l\in S\\ l\neq i}} (W_i^\top W_j)(W_j^\top A_l^*) \Bigg\}\Bigg]$$

$$e_i = \mathbb{E}_{S\in \mathbb{S}}\Bigg[\mathbf{1}_{i\in S} \times \Bigg\{ \sum_{\substack{j\in S\\ j\neq i}} \epsilon_i \epsilon_j W_j - m_1 \sum_{\substack{j,k\in S\\ j\neq i}} \epsilon_i (W_j^\top A_k^*) W_j - m_1 \sum_{\substack{j,k\in S\\ j\neq i}} \epsilon_j (W_i^\top A_k^*) W_j$$

$$+ m_2 \sum_{\substack{j,k\in S\\ j\neq i}} (W_i^\top A_k^*)(W_j^\top A_k^*) W_j + m_1^2 \sum_{\substack{j,k,l\in S\\ j\neq i\\ k\neq l}} (W_i^\top A_k^*)(W_j^\top A_l^*) W_j$$

$$- 2Dm_1^2 \sum_{\substack{j,k\in S\\ j\neq i\\ j\neq k}} (W_i^\top A_k^*) A_j^* - 2Dm_2 \sum_{\substack{j\in S\\ j\neq i}} (W_i^\top A_j^*) A_j^* + Dm_1 \sum_{\substack{j\in S\\ j\neq i}} \epsilon_i A_j^*$$

$$- m_1 \sum_{\substack{j,k\in S\\ k\neq i}} \epsilon_j (W_i^\top W_j) A_k^* + m_2 \sum_{\substack{j,k\in S\\ k\neq i}} (W_i^\top W_j)(W_j^\top A_k^*) A_k^* + m_1^2 \sum_{\substack{j,k,l\in S\\ k\neq i,l}} (W_i^\top W_j)(W_j^\top A_l^*) A_k^* \Bigg\}\Bigg]$$

We will now estimate bounds on each of the terms α_i, β_i, $||e_i||$. We will separate them as $\alpha_i = \tilde{\alpha}_i + \hat{\alpha}_i$ (similarly for the other terms). Where the tilde terms are those that come as a coefficient of m_2, and the hat terms are the ones that come as coefficient of m_1 or ϵ or both.

2.9.2 Estimating the m_2 dependent parts of the derivative

Since $||A_i^*|| = 1$ and W_i is being assumed to be within a $0 < \delta < 1$ ball of A_i^* we can use the following inequalities:

$$||W_i|| = ||W_i - A_i^* + A_i^*|| \leq ||W_i - A_i^*|| + ||A_i^*|| = \delta + 1$$

$$||W_i|| \geq 1 - \delta$$

$$\langle W_i, A_i^* \rangle = \langle W_i - A_i^*, A_i^* \rangle + \langle A_i^*, A_i^* \rangle \leq ||W_i - A_i^*|| ||A_i^*|| + 1 \leq \delta + 1$$

$$\langle W_i, A_i^* \rangle \geq 1 - \delta$$

$$|\langle W_j, A_i^* \rangle| = |\langle W_j - A_j^*, A_i^* \rangle + \langle A_j^*, A_i^* \rangle| \leq \frac{\mu}{\sqrt{n}} + ||W_j - A_j^*|| ||A_i^*|| = \frac{\mu}{\sqrt{n}} + \delta$$

$$|\langle W_i, W_j \rangle| = |\langle W_i - A_i^*, W_j \rangle + \langle A_i^*, W_j \rangle| \leq \delta(1+\delta) + (\delta + \frac{\mu}{\sqrt{n}}) = \delta^2 + 2\delta + \frac{\mu}{\sqrt{n}}$$

$$\langle W_i, W_i \rangle = ||W_i||^2 \geq (1-\delta)^2$$

$$\langle W_i, W_i \rangle = ||W_i||^2 \leq (1+\delta)^2$$

Bounding $\tilde{\beta}_i$

$$\tilde{\beta}_i = \mathbb{E}_{S \in \mathbb{S}} \left[\mathbf{1}_{i \in S} \left\{ 2Dm_2(W_i^\top A_i^*) - m_2 \sum_{j \in S} (W_i^\top W_j)(W_j^\top A_i^*) \right\} \right]$$

$$= \mathbb{E}_{S \in \mathbb{S}} \left[\mathbf{1}_{i \in S} \left\{ 2Dm_2 \langle W_i, A_i^* \rangle - m_2 ||W_i||^2 \langle W_i, A_i^* \rangle - m_2 \sum_{\substack{j \in S \\ j \neq i}} \langle W_i, W_j \rangle \langle W_j, A_i^* \rangle \right\} \right]$$

Evaluating the outer expectation we get,

$$\tilde{\beta}_i = \sum_{\{S \in \mathbb{S}: i \in S\}} q_S 2Dm_2 \langle W_i, A_i^* \rangle - \sum_{\{S \in \mathbb{S}: i \in S\}} q_S m_2 ||W_i||^2 \langle W_i, A_i^* \rangle$$

$$- m_2 \sum_{\substack{j=1 \\ j \neq i}}^{h} \langle W_i, W_j \rangle \langle W_j, A_i^* \rangle \sum_{\{S \in \mathbb{S}: i, j \in S, i \neq j\}} q_S \tag{2.5}$$

$$= 2Dq_i m_2 \langle W_i, A_i^* \rangle - q_i m_2 ||W_i||^2 \langle W_i, A_i^* \rangle - m_2 \sum_{\substack{j=1 \\ j \neq i}}^{h} q_{ij} \langle W_i, W_j \rangle \langle W_j, A_i^* \rangle$$

Upper bounding the above we get,

$$\tilde{\beta}_i \leq 2Dm_2 h^{p-1}(1+\delta) - m_2 h^{p-1}(1-\delta)^3 + m_2 h^{2p-1}\left(\delta + \frac{\mu}{\sqrt{n}}\right)\left(\delta^2 + 2\delta + \frac{\mu}{\sqrt{n}}\right)$$

$$= 2Dm_2 h^{p-1}(1 + h^{-p-\nu^2}) - m_2 h^{p-1}(1 - 3h^{-p-\nu^2} + 3h^{-2p-2\nu^2} - h^{-3p-3\nu^2})$$

$$+ m_2 h^{2p-1}(h^{-3p-3\nu^2} + 2h^{-2p-2\nu^2} + h^{-2p-2\nu^2-\xi} + 3h^{-p-\nu^2-\xi} + h^{-2\xi}) \tag{2.6}$$

Similarly for the lower bound on β_i we get,

$$\tilde{\beta}_i \geq 2Dm_2 h^{p-1}(1-\delta) - m_2 h^{p-1}(1+\delta)^3 - m_2 h^{2p-1}\left(\delta + \frac{\mu}{\sqrt{n}}\right)\left(\delta^2 + 2\delta + \frac{\mu}{\sqrt{n}}\right)$$

$$= 2Dm_2 h^{p-1}(1 - h^{-p-\nu^2}) - m_2 h^{p-1}(1 + 3h^{-p-\nu^2} + 3h^{-2p-2\nu^2} + h^{-3p-3\nu^2})$$

$$- m_2 h^{2p-1}(h^{-3p-3\nu^2} + 2h^{-2p-2\nu^2} + h^{-2p-2\nu^2-\xi} + 3h^{-p-\nu^2-\xi} + h^{-2\xi}) \tag{2.7}$$

Thus for $0 < p < 2\xi$ and $D = 1$, we have $\beta = \Theta\left(m_2 h^{p-1}\right)$

Bounding $\tilde{\alpha_i}$

$$\begin{aligned}
\tilde{\alpha_i} &= \mathbb{E}_{S\in\mathbb{S}}\left[\mathbf{1}_{i\in S}\left\{m_2\sum_{k\in S}(W_i^\top A_k^*)^2\right\}\right] \\
&= \mathbb{E}_{S\in\mathbb{S}}\left[\mathbf{1}_{i\in S}\left\{m_2\langle W_i, A_i^*\rangle^2 + m_2\sum_{\substack{k\in S\\k\neq i}}\langle W_i, A_k^*\rangle^2\right\}\right] \\
&= \sum_{\{S\in\mathbb{S}:i\in S\}} m_2\langle W_i, A_i^*\rangle^2 q_S + \sum_{\substack{k=1\\k\neq i}}^{h}\sum_{\{S\in\mathbb{S}:i,k\in S\}}\langle W_i, A_k^*\rangle^2 q_S \\
&= m_2\langle W_i, A_i^*\rangle^2\sum_{\{S\in\mathbb{S}:i\in S\}} q_S + m_2\sum_{\substack{k=1\\k\neq i}}^{h}\langle W_i, A_k^*\rangle^2\left(\sum_{\{S\in\mathbb{S}:i,k\in S,i\neq k\}} q_S\right) \\
&= q_i m_2\langle W_i, A_i^*\rangle^2 + m_2\sum_{\substack{k=1\\k\neq i}}^{h} q_{ik}\langle W_i, A_k^*\rangle^2 \\
&= h^{p-1}m_2\langle W_i, A_i^*\rangle^2 + m_2 h^{2p-1}\max\,\langle W_i, A_k^*\rangle^2
\end{aligned}$$

The above implies the following bounds,

$$h^{p-1}m_2(1-h^{-p-\nu^2})^2 \leq \tilde{\alpha_i} \leq h^{p-1}m_2(1+h^{-p-\nu^2})^2 + m_2h^{2p-1}(h^{-p-\nu^2}+h^{-\xi})^2 \tag{2.8}$$

As long as $0 < p < 2\xi$, $\tilde{\alpha_i} = \Theta\left(m_2 h^{p-1}\right)$

Bounding $||\tilde{e}_i||_2$

$$\tilde{e}_i = \mathbb{E}_{S\in\mathbb{S}}\left[\mathbf{1}_{i\in S}\times\left\{m_2\sum_{\substack{j,k\in S\\ j\neq i}}(W_i^\top A_k^*)(W_j^\top A_k^*)W_j + (-2D)m_2\sum_{\substack{j\in S\\ j\neq i}}(W_i^\top A_j^*)A_j^*\right\}\right]$$

$$+\mathbb{E}_{S\in\mathbb{S}}\left[\mathbf{1}_{i\in S}\times\left\{m_2\sum_{\substack{j,k\in S\\ k\neq i}}(W_i^\top W_j)(W_j^\top A_k^*)A_k^*\right\}\right]$$

$$= \mathbb{E}_{S\in\mathbb{S}}\Bigg[\mathbf{1}_{i\in S}\times m_2\Bigg\{\sum_{j(=k)\in S\setminus i}(W_i^\top A_j^*)(W_j^\top A_j^*)W_j + \sum_{\substack{j\in S\setminus i\\ k\in S\setminus i,j}}(W_i^\top A_k^*)(W_j^\top A_k^*)W_j$$

$$+\sum_{\substack{j\in S\setminus i\\ k=i}}(W_i^\top A_i^*)(W_j^\top A_i^*)W_j\Bigg\}\Bigg]$$

$$+\mathbb{E}_{S\in\mathbb{S}}\left[\mathbf{1}_{i\in S}\times(-2D)m_2\left\{\sum_{\substack{j\in S\\ j\neq i}}(W_i^\top A_j^*)A_j^*\right\}\right]$$

$$+\mathbb{E}_{S\in\mathbb{S}}\Bigg[\mathbf{1}_{i\in S}\times m_2\Bigg\{\sum_{k(=j)\in S\setminus i}(W_i^\top W_k)(W_k^\top A_k^*)A_k^* + \sum_{\substack{k\in S\setminus i\\ j\in S\setminus i,k}}(W_i^\top W_j)(W_j^\top A_k^*)A_k^*$$

$$+\sum_{\substack{k\in S\setminus i\\ j=i}}(W_i^\top W_i)(W_i^\top A_k^*)A_k^*\Bigg\}\Bigg]$$

$$
\begin{aligned}
\tilde{e}_i = m_2 \Bigg\{ & \sum_{j=1, j\neq i}^{h} (W_i^\top A_j^*)(W_j^\top A_j^*) W_j \sum_{\{S \in \mathbb{S}: i,j \in S, i \neq j\}} q_S \\
& + \sum_{\substack{j,k=1 \\ j\neq k \neq i}}^{h} (W_i^\top A_k^*)(W_j^\top A_k^*) W_j \sum_{\{S \in \mathbb{S}: i,j,k \in S, i \neq j \neq k\}} q_S \\
& + \sum_{\substack{j=1 \\ j\neq i}}^{h} (W_i^\top A_i^*)(W_j^\top A_i^*) W_j \sum_{\{S \in \mathbb{S}: i,j \in S, i \neq j\}} q_S \Bigg\} \\
& + (-2D) m_2 \left\{ \sum_{\substack{j=1 \\ j\neq i}}^{h} (W_i^\top A_j^*) A_j^* \sum_{\{S \in \mathbb{S}: i,j \in S, i \neq j\}} q_S \right\} \\
& + m_2 \Bigg\{ \sum_{\substack{k=1 \\ k\neq i}}^{h} (W_i^\top W_k)(W_k^\top A_k^*) A_k^* \sum_{\{S \in \mathbb{S}: i,k \in S, i \neq k\}} q_S \\
& + \sum_{\substack{j,k=1 \\ j\neq i \neq k}}^{h} (W_i^\top W_j)(W_j^\top A_k^*) A_k^* \sum_{\{S \in \mathbb{S}: i,j,k \in S, i \neq j \neq k\}} q_S \\
& + \sum_{\substack{k=1 \\ k\neq i}}^{h} (W_i^\top W_i)(W_i^\top A_k^*) A_k^* \sum_{\{S \in \mathbb{S}: i,k \in S, i \neq k\}} q_S \Bigg\}
\end{aligned}
$$

$$\tilde{e}_i = m_2 \Bigg\{ \sum_{j=1, j\neq i}^{h} q_{ij} (W_i^\top A_j^*)(W_j^\top A_j^*) W_j + \sum_{\substack{j,k=1 \\ j\neq k \neq i}}^{h} q_{ijk} (W_i^\top A_k^*)(W_j^\top A_k^*) W_j$$

$$+ \sum_{\substack{j=1 \\ j\neq i}}^{h} q_{ij} (W_i^\top A_i^*)(W_j^\top A_i^*) W_j \Bigg\} + (-2D) m_2 \left\{ \sum_{\substack{j=1 \\ j\neq i}}^{h} q_{ij} (W_i^\top A_j^*) A_j^* \right\}$$

$$+ m_2 \Bigg\{ \sum_{\substack{k=1 \\ k\neq i}}^{h} q_{ik} (W_i^\top W_k)(W_k^\top A_k^*) A_k^* + \sum_{\substack{j,k=1 \\ j\neq i \neq k}}^{h} q_{ijk} (W_i^\top W_j)(W_j^\top A_k^*) A_k^*$$

$$+ \sum_{\substack{k=1 \\ k\neq i}}^{h} q_{ik} (W_i^\top W_i)(W_i^\top A_k^*) A_k^* \Bigg\}$$

Upper bounding the norm of this vector $\tilde{e}_i$ we get,

$$\begin{aligned}
||\tilde{e}_i|| \le\ & m_2 h^{2p-1}\left(\delta + \frac{\mu}{\sqrt{n}}\right)(1+\delta)^2 + m_2 h^{3p-1}\left(\delta + \frac{\mu}{\sqrt{n}}\right)^2 (1+\delta) \\
& + m_2 h^{2p-1}\left(\delta + \frac{\mu}{\sqrt{n}}\right)(1+\delta)^2 + 2D m_2 h^{2p-1}\left(\delta + \frac{\mu}{\sqrt{n}}\right) \\
& + m_2 h^{2p-1}\left(\delta^2 + 2\delta + \frac{\mu}{\sqrt{n}}\right)(1+\delta) + m_2 h^{3p-1}\left(\delta^2 + 2\delta + \frac{\mu}{\sqrt{n}}\right)\left(\delta + \frac{\mu}{\sqrt{n}}\right) \\
& + m_2 h^{2p-1}\left(\delta + \frac{\mu}{\sqrt{n}}\right)(1+\delta)^2 \\
& \le m_2 h^{2p-1}(h^{-p-\nu^2} + 2h^{-2p-2\nu^2} + h^{-3p-3\nu^2} + 2h^{-p-\nu^2-\xi} + h^{-2p-2\nu^2-\xi} + h^{-\xi}) \\
& + m_2 h^{3p-1}(h^{-2p-2\nu^2} + h^{-3p-3\nu^2} + 2h^{-p-\nu^2-\xi} + 2h^{-2p-2\nu^2-\xi} + h^{-2\xi} + h^{-p-\nu^2-2\xi}) \\
& + m_2 h^{2p-1}(h^{-p-\nu^2} + 2h^{-2p-2\nu^2} + h^{-3p-3\nu^2} + 2h^{-p-\nu^2-\xi} + h^{-2p-2\nu^2-\xi} + h^{-\xi}) \\
& + 2D m_2 h^{2p-1}(h^{-p-\nu^2} + h^{-\xi}) \\
& + m_2 h^{2p-1}(2h^{-p-\nu^2} + 3h^{-2p-2\nu^2} + h^{-3p-3\nu^2} + h^{-p-\nu^2-\xi} + h^{-\xi}) \\
& + m_2 h^{3p-1}(2h^{-2p-2\nu^2} + h^{-3p-3\nu^2} + 3h^{-p-\nu^2-\xi} + h^{-2p-2\nu^2-\xi} + h^{-2\xi}) \\
& + m_2 h^{2p-1}(h^{-p-\nu^2} + 2h^{-2p-2\nu^2} + h^{-3p-3\nu^2} + 2h^{-p-\nu^2-\xi} + h^{-2p-2\nu^2-\xi} + h^{-\xi})
\end{aligned} \tag{2.9}$$

If $D = 1$ and $0 < p < \xi$, we get $||\tilde{e}_i|| = o(m_2 h^{p-1})$

2.9.3 Estimating the m_1 dependent parts of the derivative

We continue working in the same regime for the W matrix as in the previous subsection. Hence the same inequalities as listed at the beginning of the previous subsection continue to hold and we use them to get the following bounds,

Bounding $\hat{\alpha}_i$

$$\hat{\alpha}_i = \mathbb{E}_{S \in \mathbb{S}}\left[\mathbf{1}_{i\in S} \times \left\{ m_1^2 \sum_{\substack{k,l \in S \\ k \neq l}} (W_i^\top A_k^*)(W_i^\top A_l^*) - 2m_1 \sum_{k \in S} \epsilon_i (W_i^\top A_k^*) + \epsilon_i^2 \right\}\right]$$

$$= \mathbb{E}_{S \in \mathbb{S}}\left[\mathbf{1}_{i\in S} \times \left\{ m_1^2 \sum_{\substack{k \in S \\ k \neq i}} \langle W_i, A_k^* \rangle \langle W_i, A_i^* \rangle + m_1^2 \sum_{\substack{l \in S \\ l \neq i}} \langle W_i, A_i^* \rangle \langle W_i, A_l^* \rangle \right.\right.$$

$$+ m_1^2 \sum_{\substack{k,l \in S \\ k \neq l \\ k \neq i \\ l \neq i}} \langle W_i, A_k^* \rangle \langle W_i, A_l^* \rangle$$

$$\left.\left. - 2m_1 \epsilon_i \langle W_i, A_i^* \rangle - 2m_1 \sum_{\substack{k \in S \\ k \neq i}} \epsilon_i \langle W_i, A_k^* \rangle + \epsilon_i^2 \right\}\right]$$

$$= 2m_1^2 \sum_{\substack{k=1 \\ k \neq i}}^{h} \langle W_i, A_k^* \rangle \langle W_i, A_i^* \rangle \sum_{\{S \in \mathbb{S} : i,k \in S, k \neq i\}} q_S$$

$$+ m_1^2 \sum_{\substack{k,l=1 \\ k \neq l \\ k \neq i \\ l \neq i}}^{h} \langle W_i, A_k^* \rangle \langle W_i, A_l^* \rangle \sum_{\{S \in \mathbb{S} : i,k,l \in S, k \neq i \neq l\}} q_S$$

$$- 2m_1 \epsilon_i \langle W_i, A_i^* \rangle \sum_{\{S \in \mathbb{S} : i \in S\}} q_S - 2m_1 \sum_{\substack{k=1 \\ k \neq i}}^{h} \epsilon_i \langle W_i, A_k^* \rangle \sum_{\{S \in \mathbb{S} : i,k \in S, k \neq i\}} q_S + \epsilon_i^2 \sum_{\{S \in \mathbb{S} : i \in S\}} q_S$$

$$\implies \hat{\alpha}_i = 2m_1^2 \sum_{\substack{k=1 \\ k \neq i}}^{h} q_{ik} \langle W_i, A_k^* \rangle \langle W_i, A_i^* \rangle + m_1^2 \sum_{\substack{k,l=1 \\ k \neq l \\ k \neq i \\ l \neq i}}^{h} q_{ikl} \langle W_i, A_k^* \rangle \langle W_i, A_l^* \rangle$$

$$- 2m_1 q_i \epsilon_i \langle W_i, A_i^* \rangle - 2m_1 \sum_{\substack{k=1 \\ k \neq i}}^{h} q_{ik} \epsilon_i \langle W_i, A_k^* \rangle + q_i \epsilon_i^2$$

We plugin $\epsilon_i = 2m_1 h^p \left(\delta + \frac{\mu}{\sqrt{n}}\right)$ for $i = 1, \dots, h$

$$\begin{aligned}
|\hat{\alpha}_i| &\leq 2m_1^2 h^{2p-1} \left(\delta + \frac{\mu}{\sqrt{n}}\right)(1+\delta) + m_1^2 h^{3p-1} \left(\delta + \frac{\mu}{\sqrt{n}}\right)^2 + 4m_1^2 h^{2p-1}(1+\delta)\left(\delta + \frac{\mu}{\sqrt{n}}\right) \\
&+ 4m_1^2 h^{3p-1} \left(\delta + \frac{\mu}{\sqrt{n}}\right)^2 + 4m_1^2 h^{3p-1} \left(\delta + \frac{\mu}{\sqrt{n}}\right)^2 \\
&= 2m_1^2 h^{2p-1}(h^{-p-\nu^2} + h^{-2p-2\nu^2} + h^{-p-\nu^2-\xi} + h^{-\xi}) \\
&+ m_1^2 h^{3p-1}(h^{-2p-2\nu^2} + 2h^{-p-\nu^2-\xi} + h^{-2\xi}) \\
&+ 4m_1^2 h^{2p-1}(h^{-p-\nu^2} + h^{-2p-2\nu^2} + h^{-\xi} + h^{-p-\nu^2-\xi}) \\
&+ 4m_1^2 h^{3p-1}(h^{-2p-2\nu^2} + 2h^{-p-\nu^2-\xi} + h^{-2\xi}) \\
&+ 4m_1^2 h^{3p-1}(h^{-2p-2\nu^2} + 2h^{-p-\nu^2-\xi} + h^{-2\xi})
\end{aligned}$$

This means that if $p < \xi$, $|\hat{\alpha}_i| = o(m_1^2 h^{p-1})$. Putting this together with the bounds obtained below 2.8, we get that $\alpha_i = \Theta(m_2 h^{p-1}) + o(m_1^2 h^{p-1})$.

Bounding $\hat{\beta}_i$

$$\hat{\beta}_i = \mathbb{E}_{S\in\mathbb{S}}\left[\mathbf{1}_{i\in S} \times \left\{2Dm_1^2 \sum_{\substack{k\in S\\k\neq i}} (W_i^\top A_k^*) - Dm_1\epsilon_i + m_1 \sum_{j\in S} \epsilon_j (W_i^\top W_j)\right.\right.$$

$$\left.\left. - m_1^2 \sum_{\substack{j,l\in S\\l\neq i}} (W_i^\top W_j)(W_j^\top A_l^*)\right\}\right]$$

$$= 2Dm_1^2 \sum_{\substack{k=1\\k\neq i}}^{h} \langle W_i, A_k^*\rangle \sum_{\{S\in\mathbb{S}:i,k\in S,k\neq i\}} q_S - Dm_1\epsilon_i \sum_{\{S\in\mathbb{S}:i\in S\}} q_S + m_1\epsilon_i||W_i||^2 \sum_{\{S\in\mathbb{S}:i\in S\}} q_S$$

$$+ m_1 \sum_{j=1,j\neq i}^{h} \epsilon_j \langle W_i, W_j\rangle \sum_{\{S\in\mathbb{S}:i,j\in S,j\neq i\}} q_S - m_1^2 \sum_{\substack{l=1\\l\neq i}}^{h} ||W_i||^2 \langle W_i, A_l^*\rangle \sum_{\{S\in\mathbb{S}:i,l\in S,l\neq i\}} q_S$$

$$- m_1^2 \sum_{\substack{l=1\\l\neq i}}^{h} \langle W_i, W_l\rangle\langle W_l, A_l^*\rangle \sum_{\{S\in\mathbb{S}:i,l\in S,l\neq i\}} q_S - m_1^2 \sum_{\substack{j,l=1\\l\neq i\\j\neq l,i}}^{h} \langle W_i, W_j\rangle\langle W_j, A_l^*\rangle \sum_{\{S\in\mathbb{S}:i,j,l\in S,l\neq i\neq i\}} q_S$$

$$= 2Dm_1^2 \sum_{\substack{k=1\\k\neq i}}^{h} q_{ik}\langle W_i, A_k^*\rangle - Dm_1\epsilon_i q_i + m_1\epsilon_i||W_i||^2 q_i + m_1 \sum_{j=1,j\neq i}^{h} \epsilon_j q_{ij}\langle W_i, W_j\rangle$$

$$- m_1^2 \sum_{\substack{l=1\\l\neq i}}^{h} ||W_i||^2\langle W_i, A_l^*\rangle q_{il} - m_1^2 \sum_{\substack{l=1\\l\neq i}}^{h} \langle W_i, W_l\rangle\langle W_l, A_l^*\rangle q_{il} - m_1^2 \sum_{\substack{j,l=1\\l\neq i\\j\neq l,i}}^{h} \langle W_i, W_j\rangle\langle W_j, A_l^*\rangle q_{ijl}$$

We plugin $\epsilon_i = 2m_1h^p\left(\delta + \frac{\mu}{\sqrt{n}}\right)$ for $i = 1, \dots, h$

$$
\begin{aligned}
|\hat{\beta}_i| &\leq 4Dm_1^2h^{2p-1}\left(\delta + \frac{\mu}{\sqrt{n}}\right) + 2m_1^2h^{2p-1}\left(\delta + \frac{\mu}{\sqrt{n}}\right)(1+\delta)^2 \\
&+ 2m_1^2h^{3p-1}\left(\delta + \frac{\mu}{\sqrt{n}}\right)\left(\delta^2 + 2\delta + \frac{\mu}{\sqrt{n}}\right) \\
&+ m_1^2h^{2p-1}(1+\delta)^2\left(\delta + \frac{\mu}{\sqrt{n}}\right) + m_1^2h^{2p-1}\left(\delta^2 + 2\delta + \frac{\mu}{\sqrt{n}}\right)(1+\delta) \\
&+ m_1^2h^{3p-1}\left(\delta^2 + 2\delta + \frac{\mu}{\sqrt{n}}\right)\left(\delta + \frac{\mu}{\sqrt{n}}\right) \\
&= 4Dm_1^2h^{2p-1}(h^{-p-\nu^2} + h^{-\xi}) \\
&+ 2m_1^2h^{2p-1}(h^{-p-\nu^2} + 2h^{-2p-2\nu^2} + h^{-3p-3\nu^2} + h^{-\xi} + 2h^{-p-\nu^2-\xi} + h^{-2p-2\nu^2-\xi}) \\
&+ 2m_1^2h^{3p-1}(2h^{-2p-2\nu^2} + h^{-3p-3\nu^2} + 3h^{-p-\nu^2-\xi} + h^{-2p-2\nu^2-\xi} + h^{-2\xi}) \\
&+ m_1^2h^{2p-1}(h^{-p-\nu^2} + 2h^{-2p-2\nu^2} + h^{-3p-3\nu^2} + h^{-\xi} + 2h^{-p-\nu^2-\xi} + h^{-2p-2\nu^2-\xi}) \\
&+ m_1^2h^{2p-1}(3h^{-2p-2\nu^2} + h^{-3p-3\nu^2} + h^{-p-\nu^2-\xi} + 2h^{-p-\nu^2} + h^{-\xi}) \\
&+ m_1^2h^{3p-1}(2h^{-2p-2\nu^2} + h^{-3p-3\nu^2} + 3h^{-p-\nu^2-\xi} + h^{-2p-2\nu^2-\xi} + h^{-2\xi})
\end{aligned}
$$

This means that if $p < \xi$, $|\hat{\beta}_i| = o(m_1^2h^{p-1})$. Putting this together with the bounds obtained below 2.5, we get that $\beta_i = \Theta(m_2h^{p-1}) + o(m_1^2h^{p-1})$.

Bounding $||\hat{e_i}||_2$

$$\hat{e_i} = \underbrace{\mathbb{E}_{S\in\mathbb{S}}\left[\mathbf{1}_{i\in S}\times\left\{\sum_{\substack{j\in S\\ j\neq i}}\epsilon_i\epsilon_j W_j - m_1\sum_{\substack{j,k\in S\\ j\neq i}}(W_j^\top A_k^*)W_j\epsilon_i - m_1\sum_{\substack{j,k\in S\\ j\neq i}}\epsilon_j(W_i^\top A_k^*)W_j\right\}\right]}_{\hat{e_{i1}}}$$

$$+\underbrace{\mathbb{E}_{S\in\mathbb{S}}\left[\mathbf{1}_{i\in S}\times\left\{m_1^2\sum_{\substack{j,k,l\in S\\ j\neq i\\ k\neq l}}(W_i^\top A_k^*)(W_j^\top A_l^*)W_j\right\}\right]}_{\hat{e_{i2}}}$$

$$+\underbrace{\mathbb{E}_{S\in\mathbb{S}}\left[\mathbf{1}_{i\in S}\times\left\{-2Dm_1^2\sum_{\substack{j,k\in S\\ j\neq i\\ k\neq i}}(W_i^\top A_k^*)A_j^* + Dm_1\sum_{\substack{j\in S\\ j\neq i}}\epsilon_i A_j^*\right\}\right]}_{\hat{e_{i3}}}$$

$$+\underbrace{\mathbb{E}_{S\in\mathbb{S}}\left[\mathbf{1}_{i\in S}\times\left\{-m_1\sum_{\substack{j,k\in S\\ k\neq i}}\epsilon_j(W_i^\top W_j)A_k^* + m_1^2\sum_{\substack{j,k,l\in S\\ k\neq i,l}}(W_i^\top W_j)(W_j^\top A_l^*)A_k^*\right\}\right]}_{\hat{e_{i4}}}$$

We estimate the different summands separately.

$$\hat{e_{i1}} = \mathbb{E}_{S \in \mathbb{S}} \left[\mathbf{1}_{i \in S} \times \left\{ \sum_{\substack{j \in S \\ j \neq i}} \epsilon_i \epsilon_j W_j \right\} \right]$$

$$+ \mathbb{E}_{S \in \mathbb{S}} \left[\mathbf{1}_{i \in S} \times (-m_1) \left\{ \sum_{j(=k) \in S \backslash i} (W_j^\top A_j^*) W_j \epsilon_i + \sum_{\substack{j \in S \backslash i \\ k \in S \backslash i,j}} (W_j^\top A_k^*) W_j \epsilon_i + \sum_{\substack{j \in S \backslash i \\ k = i}} (W_j^\top A_i^*) W_j \epsilon_i \right\} \right]$$

$$+ \mathbb{E}_{S \in \mathbb{S}} \left[\mathbf{1}_{i \in S} \times (-m_1) \left\{ \sum_{j(=k) \in S \backslash i} \epsilon_j (W_i^\top A_j^*) W_j + \sum_{\substack{j \in S \backslash i \\ k \in S \backslash i,j}} \epsilon_j (W_i^\top A_k^*) W_j + \sum_{\substack{j \in S \backslash i \\ k = i}} \epsilon_j (W_i^\top A_i^*) W_j \right\}\right.$$

We substitute, $\epsilon = 2m_1 h^p (h^{-p-\nu^2} + h^{-\xi})$ and for any two vectors $\mathbf{x}$ and $\mathbf{y}$ and any two scalars a and b we use the inequality, $||a\mathbf{x} + b\mathbf{y}||_2 \leq |a|_{max} ||\mathbf{x}||_{2,max} + |b|_{max} ||\mathbf{y}||_{2,max}$to get,

$$
\begin{aligned}
||\hat{e_{i1}}||_2 \leq{}& 4m_1^2 h^{2p} \left(\delta + \frac{\mu}{\sqrt{n}}\right)^2 \sum_{j=1, j\neq i}^{h} q_{ij} ||W_j|| \\
&+ 2m_1^2 h^p \left(\delta + \frac{\mu}{\sqrt{n}}\right) \left(\sum_{j=1, j\neq i}^{h} q_{ij} \langle W_j, A_j^* \rangle W_j + \sum_{j,k=1, j\neq i, k\neq i,j}^{h} q_{ijk} \langle W_j, A_k^* \rangle W_j \right. \\
&\left. + \sum_{j=1, j\neq i}^{h} q_{ij} \langle W_j, A_i^* \rangle W_j \right) \\
&+ 2m_1^2 h^p \left(\delta + \frac{\mu}{\sqrt{n}}\right) \left(\sum_{j=1, j\neq i}^{h} q_{ij} \langle W_i, A_j^* \rangle W_j + \sum_{j,k=1, j\neq i, k\neq i,j}^{h} q_{ijk} \langle W_i, A_k^* \rangle W_j \right. \\
&\left. + \sum_{j=1, j\neq i}^{h} q_{ij} \langle W_i, A_i^* \rangle W_j \right) \\
\implies ||\hat{e_{i1}}||_2 \leq{}& 4m_1^2 h^{2p} h^{2p-1} (1+\delta) \left(\delta + \frac{\mu}{\sqrt{n}}\right)^2 \\
&+ 2m_1^2 h^p \left(\delta + \frac{\mu}{\sqrt{n}}\right) \left(h^{2p-1} (1+\delta)^2 + h^{3p-1} \left(\delta + \frac{\mu}{\sqrt{n}}\right) (1+\delta) \right. \\
&\left. + h^{2p-1} \left(\delta + \frac{\mu}{\sqrt{n}}\right) (1+\delta) \right) \\
&+ 2m_1^2 h^p \left(\delta + \frac{\mu}{\sqrt{n}}\right) \left(h^{2p-1} \left(\delta + \frac{\mu}{\sqrt{n}}\right) (1+\delta) + h^{3p-1} \left(\delta + \frac{\mu}{\sqrt{n}}\right) (1+\delta) \right. \\
&\left. + h^{2p-1} (1+\delta)^2 \right)
\end{aligned}
$$

$$\implies ||\hat{e_{i1}}||_2 \leq 4m_1^2h^{4p-1}(1+\delta)\left(\delta+\frac{\mu}{\sqrt{n}}\right)^2$$

$$+ 2m_1^2h^{3p-1}\left(\delta+\frac{\mu}{\sqrt{n}}\right)(1+\delta)^2 + 2m_1^2h^{4p-1}\left(\delta+\frac{\mu}{\sqrt{n}}\right)^2(1+\delta)$$

$$+ 2m_1^2h^{3p-1}\left(\delta+\frac{\mu}{\sqrt{n}}\right)^2(1+\delta) + 2m_1^2h^{3p-1}\left(\delta+\frac{\mu}{\sqrt{n}}\right)^2(1+\delta)$$

$$+ 2m_1^2h^{4p-1}\left(\delta+\frac{\mu}{\sqrt{n}}\right)^2(1+\delta) + 2m_1^2h^{3p-1}\left(\delta+\frac{\mu}{\sqrt{n}}\right)(1+\delta)^2$$

$$\implies ||\hat{e_{i1}}||_2 \leq 8m_1^2h^{4p-1}(1+\delta)\left(\delta+\frac{\mu}{\sqrt{n}}\right)^2 + 4m_1^2h^{3p-1}\left(\delta+\frac{\mu}{\sqrt{n}}\right)(1+\delta)^2$$

$$+ 4m_1^2h^{3p-1}\left(\delta+\frac{\mu}{\sqrt{n}}\right)^2(1+\delta)$$

$$\implies ||\hat{e_{i1}}||_2 \leq 8m_1^2h^{4p-1}(h^{-2p-2\nu^2} + h^{-3p-3\nu^2} + 2h^{-p-\nu^2-\xi}$$

$$+ 2h^{-2p-2\nu^2-\xi} + h^{-p-\nu^2-2\xi} + h^{-2\xi})$$

$$+ 4m_1^2h^{3p-1}(h^{-p-\nu^2} + h^{-3p-3\nu^2} + 2h^{-2p-2\nu^2} + h^{-\xi} + h^{-2p-2\nu^2-\xi} + 2h^{-p-\nu^2-\xi})$$

$$+ 4m_1^2h^{3p-1}(h^{-2p-2\nu^2} + h^{-3p-3\nu^2} + 2h^{-p-\nu^2-\xi}$$

$$+ 2h^{-2p-2\nu^2-\xi} + h^{-p-\nu^2-2\xi} + h^{-2\xi})$$

$$\implies ||\hat{e_{i1}}||_2 \leq 8m_1^2 h^{p-1}(h^{p-2\nu^2} + h^{-3\nu^2} + 2h^{p-\nu^2+p-\xi} + 2h^{-2\nu^2+p-\xi} + h^{-\nu^2+2p-2\xi} + h^{3p-2\xi})$$

$$+ 4m_1^2 h^{p-1}(h^{p-\nu^2} + h^{-p-3\nu^2} + 2h^{-2\nu^2} + h^{2p-\xi} + h^{-2\nu^2-\xi} + 2h^{-\nu^2+p-\xi})$$

$$+ 4m_1^2 h^{p-1}(h^{-2\nu^2} + h^{-p-3\nu^2} + 2h^{-\nu^2+p-\xi} + 2h^{-2\nu^2-\xi} + h^{-\nu^2+p-2\xi} + h^{2p-2\xi})$$

From the above it follows that, $||\hat{e_{i1}}||_2 = o(m_1^2 h^{p-1})$ for $p < \nu^2$ and $2p < \xi$.

$$\hat{e_{i2}} = \mathbb{E}_{S \in \mathbb{S}}\left[\mathbf{1}_{i \in S} \times m_1^2 \left\{ \sum_{\substack{j,k,l \in S \\ j \neq i \\ k \neq l}} (W_i^\top A_k^*)(W_j^\top A_l^*) W_j \right\}\right]$$

$$= \mathbb{E}_{S \in \mathbb{S}}\Bigg[\mathbf{1}_{i \in S} \times m_1^2 \Bigg\{ \sum_{\substack{j \in S \\ j \neq i}} (W_i^\top A_j^*)(W_j^\top A_i^*) W_j + \sum_{\substack{j,k \in S \\ k \neq j \neq i}} (W_i^\top A_k^*)(W_j^\top A_i^*) W_j$$

$$+ \sum_{\substack{j \in S \\ j \neq i}} (W_i^\top A_i^*)(W_j^\top A_j^*) W_j + \sum_{\substack{j,l \in S \\ l \neq j \neq i}} (W_i^\top A_i^*)(W_j^\top A_l^*) W_j$$

$$+ \sum_{\substack{j,l \in S \\ l \neq j \neq i}} (W_i^\top A_j^*)(W_j^\top A_l^*) W_j + \sum_{\substack{j,k \in S \\ k \neq j \neq i}} (W_i^\top A_k^*)(W_j^\top A_j^*) W_j$$

$$+ \sum_{\substack{j,k,l \in S \\ l \neq k \neq j \neq i}} (W_i^\top A_k^*)(W_j^\top A_l^*) W_j \Bigg\}\Bigg]$$

$$\implies \hat{e_{i2}} = m_1^2 \Bigg\{ \sum_{\substack{j=1\\ j\neq i}}^{h} q_{ij}(W_i^\top A_j^*)(W_j^\top A_i^*)W_j + \sum_{\substack{j,k=1\\ k\neq j\neq i}}^{h} q_{ijk}(W_i^\top A_k^*)(W_j^\top A_i^*)W_j$$

$$+ \underbrace{\sum_{\substack{j=1\\ j\neq i}}^{h} q_{ij}(W_i^\top A_i^*)(W_j^\top A_j^*)W_j}_{\mathbf{a}}$$

$$+ \sum_{\substack{j,l=1\\ l\neq j\neq i}}^{h} q_{ijl}(W_i^\top A_i^*)(W_j^\top A_l^*)W_j + \sum_{\substack{j,l=1\\ l\neq j\neq i}}^{h} q_{ijl}(W_i^\top A_j^*)(W_j^\top A_l^*)W_j$$

$$+ \sum_{\substack{j,k=1\\ k\neq j\neq i}}^{h} q_{ijk}(W_i^\top A_k^*)(W_j^\top A_j^*)W_j + \sum_{\substack{j,k,l\in S\\ l\neq k\neq j\neq i}} q_{ijkl}(W_i^\top A_k^*)(W_j^\top A_l^*)W_j \Bigg\}$$

$$\implies ||\hat{e_{i2}}|| \leq m_1^2 \Bigg\{ h^{2p-1} \left(\delta + \frac{\mu}{\sqrt{n}}\right)^2 (1+\delta) + h^{3p-1} \left(\delta + \frac{\mu}{\sqrt{n}}\right)^2 (1+\delta) + ||\mathbf{a}||$$

$$+ h^{3p-1} \left(\delta + \frac{\mu}{\sqrt{n}}\right) (1+\delta)^2 + h^{3p-1} \left(\delta + \frac{\mu}{\sqrt{n}}\right)^2 (1+\delta)$$

$$+ h^{3p-1} \left(\delta + \frac{\mu}{\sqrt{n}}\right) (1+\delta)^2 + h^{4p-1} \left(\delta + \frac{\mu}{\sqrt{n}}\right)^2 (1+\delta) \Bigg\}$$

$$\implies ||\hat{e_{i2}}|| \leq m_1^2 \Bigg\{ h^{2p-1} (h^{-2p-2\nu^2} + h^{-3p-3\nu^2} + 2h^{-p-\nu^2-\xi}$$

$$+ 2h^{-2p-2\nu^2-\xi} + h^{-p-\nu^2-2\xi} + h^{-2\xi}) + h^{3p-1}(h^{-2p-2\nu^2} + h^{-3p-3\nu^2}$$

$$+ 2h^{-p-\nu^2-\xi} + 2h^{-2p-2\nu^2-\xi} + h^{-p-\nu^2-2\xi} + h^{-2\xi})$$

$$+ ||\mathbf{a}||$$

$$+ h^{3p-1}(h^{-p-\nu^2} + h^{-3p-3\nu^2} + 2h^{-2p-2\nu^2} + h^{-2p-2\nu^2-\xi} + 2h^{-p-\nu^2-\xi} + h^{-\xi})$$

$$+ h^{3p-1}(h^{-2p-2\nu^2} + h^{-3p-3\nu^2} + 2h^{-p-\nu^2-\xi} + 2h^{-2p-2\nu^2-\xi} + h^{-p-\nu^2-2\xi} + h^{-2\xi})$$

$$+ h^{3p-1}(h^{-p-\nu^2} + h^{-3p-3\nu^2} + 2h^{-2p-2\nu^2} + h^{-2p-2\nu^2-\xi} + 2h^{-p-\nu^2-\xi} + h^{-\xi})$$

$$+ h^{4p-1}(h^{-2p-2\nu^2} + h^{-3p-3\nu^2} + 2h^{-p-\nu^2-\xi} + 2h^{-2p-2\nu^2-\xi} + h^{-p-\nu^2-2\xi} + h^{-2\xi}) \Bigg\}$$

$$\implies ||\hat{e_{i2}}|| \leq m_1^2 \Big\{ h^{p-1}(h^{-p-2\nu^2} + h^{-2p-3\nu^2} + 2h^{-\nu^2-\xi} + 2h^{-p-2\nu^2-\xi} + h^{-\nu^2-2\xi} + h^{p-2\xi})$$

$$+ h^{p-1}(h^{-2\nu^2} + h^{-p-3\nu^2} + 2h^{-\nu^2+p-\xi} + 2h^{-2\nu^2-\xi} + h^{-\nu^2+p-2\xi} + h^{2p-2\xi})$$

$$+ ||\mathbf{a}||$$

$$+ h^{p-1}(h^{p-\nu^2} + h^{-p-3\nu^2} + 2h^{-2\nu^2} + h^{-2\nu^2-\xi} + 2h^{-\nu^2+p-\xi} + h^{2p-\xi})$$

$$+ h^{p-1}(h^{-2\nu^2} + h^{-p-3\nu^2} + 2h^{-\nu^2+p-\xi} + 2h^{-2\nu^2-\xi} + h^{-\nu^2+p-2\xi} + h^{2p-2\xi})$$

$$+ h^{p-1}(h^{p-\nu^2} + h^{-2p-3\nu^2} + 2h^{-2\nu^2} + h^{-2\nu^2-\xi} + 2h^{-\nu^2+p-\xi} + h^{2p-\xi})$$

$$+ h^{p-1}(h^{p-2\nu^2} + h^{-3\nu^2} + 2h^{p-\nu^2+p-\xi} + 2h^{-2\nu^2+p-\xi} + h^{-\nu^2+2p-2\xi} + h^{3p-2\xi}) \Big\}$$

Now let us find a bound for $||\mathbf{a}||$.

$$\mathbf{a} = \sum_{\substack{j=1 \\ j\neq i}}^{h} q_{ij}(W_i^\top A_i^*)(W_j^\top A_j^*)W_j$$

$$= \langle W_i, A_i^* \rangle q_{ij} W_{-j}^\top \text{diag}(W_{-j} A_{-j}^*)$$

Where A_{-j}^* is the dictionary A^* with the jth column set to zero, W_{-j} is the dictionary W with the jth row set to zero, and $\text{diag}(W_{-j}A_{-j}^*)$ is the h-dimensional vector containing the diagonal elements of the matrix $W_{-j}A_{-j}^*$. We also make use of the distributional assumption that q_{ij} is the same for all i, j in order to pull q_{ij} out of the sum.

$$\begin{aligned}
||\mathbf{a}||_2 &= h^{2p-2}\langle W_i, A_i^*\rangle ||W_{-j}^\top \mathrm{diag}(W_{-j}A_{-j}^*)||_2 \\
&\leq h^{2p-2}(1+\delta)||W_{-j}^\top||_2||\mathrm{diag}(W_{-j}A_{-j}^*)||_2 \\
&\leq h^{2p-2}(1+\delta)^2 h^{1/2}\sqrt{\lambda_{\max}(W_{-j}^\top W_{-j})} \\
&\leq h^{2p-2}(1+\delta)^2 h^{1/2}\sqrt{h\left(\delta^2+2\delta+\frac{\mu}{\sqrt{n}}\right)+(1+\delta)^2} \\
&= h^{p-1}\sqrt{h^{2p-2}\times h\times(1+\delta)^4\times\left(h\left(\delta^2+2\delta+\frac{\mu}{\sqrt{n}}\right)+(1+\delta)^2\right)} \\
&= h^{p-1}\sqrt{h^{2p-1}\times(1+h^{-p-\nu^2})^4\times\left(h(h^{-2p-2\nu^2}+2h^{-p-\nu^2}+h^{-\xi})+(1+h^{-p-\nu^2})^2\right)} \\
&= h^{p-1}\sqrt{(1+h^{-p-\nu^2})^4\times(h^{-2\nu^2}+2h^{p-\nu^2}+h^{2p-\xi}+h^{2p-1}(1+h^{-p-\nu^2})^2)}
\end{aligned}$$

Here $||W_{-j}^\top||_2$ is the spectral norm of $W_{-j}^\top$, and is the top singular value of the matrix. We use Gershgorin's Circle theorem to bound the top eigenvalue of $W_{-j}^\top W_{-j}$ by its maximum row sum.

If $p < \frac{\xi}{2}$, $p < \frac{1}{2}$, and $p < \nu^2$, then $||\hat{e_{i2}}|| = o(m_1^2 h^{p-1})$

$$\hat{e_{i3}} = \mathbb{E}_{S \in \mathbb{S}} \left[\mathbf{1}_{i \in S} \times \left\{ Dm_1 \sum_{\substack{j \in S \\ j \neq i}} \epsilon_i A_j^* - 2Dm_1^2 \sum_{\substack{j,k \in S \\ j \neq i \\ k \neq i}} (W_i^\top A_k^*) A_j^* \right\} \right]$$

$$= \mathbb{E}_{S \in \mathbb{S}} \left[\mathbf{1}_{i \in S} \times \left\{ Dm_1 \sum_{\substack{j \in S \\ j \neq i}} \epsilon_i A_j^* - 2Dm_1^2 \sum_{\substack{j \in S \\ j \neq i}} (W_i^\top A_j^*) A_j^* - 2Dm_1^2 \sum_{\substack{j,k \in S \\ k \neq j \neq i}} (W_i^\top A_k^*) A_j^* \right\} \right]$$

$$= Dm_1 \sum_{\substack{j=1 \\ j \neq i}}^{h} \epsilon_i A_j^* \sum_{\{S \in \mathbb{S}: i,j \in S, i \neq j\}} q_S - 2Dm_1^2 \sum_{\substack{j=1 \\ j \neq i}}^{h} (W_i^\top A_j^*) A_j^* \sum_{\{S \in \mathbb{S}: i,j \in S, i \neq j\}} q_S$$

$$- 2Dm_1^2 \sum_{\substack{j,k=1 \\ k \neq j \neq i}}^{h} (W_i^\top A_k^*) A_j^* \sum_{\{S \in \mathbb{S}: i,j,k \in S, i \neq j \neq k\}} q_S$$

$$= Dm_1 \sum_{\substack{j=1 \\ j \neq i}}^{h} q_{ij} \epsilon_i A_j^* - 2Dm_1^2 \sum_{\substack{j=1 \\ j \neq i}}^{h} q_{ij} (W_i^\top A_j^*) A_j^* - 2Dm_1^2 \sum_{\substack{j,k=1 \\ k \neq j \neq i}}^{h} q_{ijk} (W_i^\top A_k^*) A_j^*$$

We plugin $\epsilon_i = 2m_1 h^p \left(\delta + \frac{\mu}{\sqrt{n}} \right)$ for $i = 1, \dots, h$

$$||\hat{e_{i3}}|| \leq 2Dm_1^2 h^{3p-1} \left(\delta + \frac{\mu}{\sqrt{n}} \right) + 2Dm_1^2 h^{2p-1} \left(\delta + \frac{\mu}{\sqrt{n}} \right) + 2Dm_1^2 h^{3p-1} \left(\delta + \frac{\mu}{\sqrt{n}} \right)$$

$$= 4Dm_1^2 h^{3p-1} (h^{-p-\nu^2} + h^{-\xi}) + 2Dm_1^2 h^{2p-1} (h^{-p-\nu^2} + h^{-\xi})$$

$$= 4Dm_1^2 h^{p-1} (h^{p-\nu^2} + h^{2p-\xi}) + 2Dm_1^2 h^{p-1} (h^{-\nu^2} + h^{p-\xi})$$

This means for $D = 1$, $p < \nu^2$ and $p < \frac{\xi}{2}$, we have $||\hat{e_{i3}}|| = o(m_1^2 h^{p-1})$

$$\hat{e_{i4}} = \mathbb{E}_{S\in\mathbb{S}}\left[\mathbf{1}_{i\in S}\times\left\{-m_1\sum_{\substack{j,k\in S\\ k\neq i}}\epsilon_j(W_i^\top W_j)A_k^* + m_1^2\sum_{\substack{j,k,l\in S\\ k\neq i,l}}(W_i^\top W_j)(W_j^\top A_l^*)A_k^*\right\}\right]$$

$$= \mathbb{E}_{S\in\mathbb{S}}\Bigg[\mathbf{1}_{i\in S}\times(-m_1)\Bigg\{\sum_{k(=j)\in S\setminus i}\epsilon_k(W_i^\top W_k)A_k^* + \sum_{\substack{j\in S\setminus i\\ k\in S\setminus i,j}}\epsilon_j(W_i^\top W_j)A_k^*$$

$$+\sum_{\substack{k\in S\setminus i\\ j=i}}\epsilon_j(W_i^\top W_i)A_k^*\Bigg\}\Bigg]$$

$$+\mathbb{E}_{S\in\mathbb{S}}\left[\mathbf{1}_{i\in S}\times m_1^2\left\{\sum_{\substack{j,k,l\in S\\ k\neq i,l}}(W_i^\top W_j)(W_j^\top A_l^*)A_k^*\right\}\right]$$

$$= \mathbb{E}_{S\in\mathbb{S}}\Bigg[\mathbf{1}_{i\in S}\times(-m_1)\Bigg\{\sum_{k(=j)\in S\setminus i}\epsilon_k(W_i^\top W_k)A_k^* + \sum_{\substack{j\in S\setminus i\\ k\in S\setminus i,j}}\epsilon_j(W_i^\top W_j)A_k^*$$

$$+\sum_{\substack{k\in S\setminus i\\ j=i}}\epsilon_j(W_i^\top W_i)A_k^*\Bigg\}\Bigg]$$

$$+\mathbb{E}_{S\in\mathbb{S}}\Bigg[\mathbf{1}_{i\in S}\times m_1^2\Bigg\{\sum_{\substack{k\in S\\ k\neq i}}(W_i^\top W_i)(W_i^\top A_i^*)A_k^* + \sum_{\substack{k\in S\\ k\neq i}}(W_i^\top W_k)(W_k^\top A_i^*)A_k^*$$

$$+\sum_{\substack{j,k\in S\\ j\neq k\neq i}}(W_i^\top W_j)(W_j^\top A_i^*)A_k^* + \sum_{\substack{k,l\in S\\ \neq k\neq i}}(W_i^\top W_i)(W_i^\top A_l^*)A_k^* + \sum_{\substack{k,l\in S\\ l\neq k\neq i}}(W_i^\top W_k)(W_k^\top A_l^*)A_k^*$$

$$+\sum_{\substack{k,l\in S\\ l\neq k\neq i}}(W_i^\top W_l)(W_l^\top A_l^*)A_k^* + \sum_{\substack{j,k,l\in S\\ j\neq k\neq l\neq i}}(W_i^\top W_j)(W_j^\top A_l^*)A_k^*\Bigg\}\Bigg]$$

$$\hat{e_{i4}} = (-m_1)\left\{\sum_{k=1,k\neq i}^{h} q_{ik}\epsilon_k (W_i^\top W_k)A_k^* + \sum_{\substack{j,k=1\\ j\neq k\neq i}}^{h} q_{ijk}\epsilon_j (W_i^\top W_j)A_k^* + \sum_{\substack{k=1\\ k\neq i}}^{h} q_{ik}\epsilon_i (W_i^\top W_i)A_k^*\right\}$$

$$+ m_1^2\Bigg\{ \underbrace{\sum_{\substack{k=1\\ k\neq i}}^{h} q_{ik}(W_i^\top W_i)(W_i^\top A_i^*)A_k^*}_{\mathbf{b}} + \sum_{\substack{k=1\\ k\neq i}}^{h} q_{ik}(W_i^\top W_k)(W_k^\top A_i^*)A_k^*$$

$$+ \sum_{\substack{j,k=1\\ j\neq k\neq i}}^{h} q_{ijk}(W_i^\top W_j)(W_j^\top A_i^*)A_k^* + \sum_{\substack{k,l=1\\ l\neq k\neq i}}^{h} q_{ikl}(W_i^\top W_i)(W_i^\top A_l^*)A_k^*$$

$$+ \sum_{\substack{k,l=1\\ l\neq k\neq i}}^{h} q_{ikl}(W_i^\top W_k)(W_k^\top A_l^*)A_k^* + \sum_{\substack{k,l=1\\ l\neq k\neq i}}^{h} q_{ikl}(W_i^\top W_l)(W_l^\top A_l^*)A_k^*$$

$$+ \sum_{\substack{j,k,l=1\\ j\neq k\neq l\neq i}}^{h} q_{ijkl}(W_i^\top W_j)(W_j^\top A_l^*)A_k^* \Bigg\}$$

We plugin $\epsilon_i = 2m_1 h^p\left(\delta + \frac{\mu}{\sqrt{n}}\right)$ for $i = 1,\ldots,h$ in the above to get,

$$
\begin{aligned}
||\hat{e_{i4}}|| \leq\ & 2m_1^2h^{3p-1}\left(\delta+\frac{\mu}{\sqrt{n}}\right)^2 + 2m_1^2h^{4p-1}\left(\delta+\frac{\mu}{\sqrt{n}}\right)\left(\delta^2+2\delta+\frac{\mu}{\sqrt{n}}\right) \\
& + 2m_1^2h^{3p-1}\left(\delta+\frac{\mu}{\sqrt{n}}\right)(1+\delta)^2 + m_1^2||\mathbf{b}|| + m_1^2h^{2p-1}\left(\delta+\frac{\mu}{\sqrt{n}}\right)\left(\delta^2+2\delta+\frac{\mu}{\sqrt{n}}\right) \\
& + m_1^2h^{3p-1}\left(\delta+\frac{\mu}{\sqrt{n}}\right)\left(\delta^2+2\delta+\frac{\mu}{\sqrt{n}}\right) + m_1^2h^{3p-1}(1+\delta)^2\left(\delta+\frac{\mu}{\sqrt{n}}\right) \\
& + m_1^2h^{3p-1}\left(\delta+\frac{\mu}{\sqrt{n}}\right)\left(\delta^2+2\delta+\frac{\mu}{\sqrt{n}}\right) \\
& + m_1^2h^{3p-1}(1+\delta)\left(\delta^2+2\delta+\frac{\mu}{\sqrt{n}}\right) + m_1^2h^{4p-1}\left(\delta+\frac{\mu}{\sqrt{n}}\right)\left(\delta^2+2\delta+\frac{\mu}{\sqrt{n}}\right)
\end{aligned}
$$

$$
\begin{aligned}
\implies ||\hat{e_{i4}}|| \leq\ & 2m_1^2h^{3p-1}\left(\delta+\frac{\mu}{\sqrt{n}}\right)^2 + 3m_1^2h^{4p-1}\left(\delta+\frac{\mu}{\sqrt{n}}\right)\left(\delta^2+2\delta+\frac{\mu}{\sqrt{n}}\right) \\
& + 3m_1^2h^{3p-1}\left(\delta+\frac{\mu}{\sqrt{n}}\right)(1+\delta)^2 + m_1^2||\mathbf{b}|| + m_1^2h^{2p-1}\left(\delta+\frac{\mu}{\sqrt{n}}\right)\left(\delta^2+2\delta+\frac{\mu}{\sqrt{n}}\right) \\
& + 2m_1^2h^{3p-1}\left(\delta+\frac{\mu}{\sqrt{n}}\right)\left(\delta^2+2\delta+\frac{\mu}{\sqrt{n}}\right) + m_1^2h^{3p-1}(1+\delta)\left(\delta^2+2\delta+\frac{\mu}{\sqrt{n}}\right)
\end{aligned}
$$

$$\implies ||\hat{e_{i4}}|| \leq 2m_1^2h^{3p-1}(h^{-2p-2\nu^2} + 2h^{-p-\nu^2-\xi} + h^{-2\xi})$$

$$+ 3m_1^2h^{4p-1}(h^{-3p-3\nu^2} + 2h^{-2p-2\nu^2} + 3h^{-p-\nu^2-\xi} + h^{-2p-2\nu^2-\xi} + h^{-2\xi})$$

$$+ 3m_1^2h^{3p-1}(h^{-3p-3\nu^2} + 2h^{-2p-2\nu^2} + 2h^{-p-\nu^2-\xi} + h^{-2p-2\nu^2-\xi} + h^{-\xi} + h^{-p-\nu^2})$$

$$+ m_1^2||\mathbf{b}||$$

$$+ m_1^2h^{2p-1}(h^{-3p-3\nu^2} + 2h^{-2p-2\nu^2} + 3h^{-p-\nu^2-\xi} + h^{-2p-2\nu^2-\xi} + h^{-2\xi})$$

$$+ 2m_1^2h^{3p-1}(h^{-3p-3\nu^2} + 2h^{-2p-2\nu^2} + 3h^{-p-\nu^2-\xi} + h^{-2p-2\nu^2-\xi} + h^{-2\xi})$$

$$+ m_1^2h^{3p-1}(h^{-3p-3\nu^2} + 3h^{-2p-2\nu^2} + h^{-p-\nu^2-\xi} + h^{-\xi} + 2h^{-p-\nu^2})$$

$$\implies ||\hat{e_{i4}}|| \leq 2m_1^2h^{p-1}(h^{-2\nu^2} + 2h^{-\nu^2+p-\xi} + h^{2p-2\xi})$$

$$+ 3m_1^2h^{p-1}(h^{-3\nu^2} + 2h^{-p-2\nu^2} + 3h^{p-\nu^2+p-\xi} + h^{-2\nu^2+p-\xi} + h^{3p-2\xi})$$

$$+ 3m_1^2h^{p-1}(h^{-p-3\nu^2} + 2h^{-2\nu^2} + 2h^{-\nu^2+p-\xi} + h^{-2\nu^2-\xi} + h^{2p-\xi} + h^{p-\nu^2})$$

$$+ m_1^2||\mathbf{b}||$$

$$+ m_1^2h^{p-1}(h^{-2p-3\nu^2} + 2h^{-p-2\nu^2} + 3h^{-\nu^2-\xi} + h^{-p-2\nu^2-\xi} + h^{p-2\xi})$$

$$+ 2m_1^2h^{p-1}(h^{-p-3\nu^2} + 2h^{-2\nu^2} + 3h^{-\nu^2+p-\xi} + h^{-2\nu^2-\xi} + h^{2p-2\xi})$$

$$+ m_1^2h^{p-1}(h^{-p-3\nu^2} + 3h^{-2\nu^2} + h^{-\nu^2+p-\xi} + h^{2p-\xi} + 2h^{p-\nu^2})$$

Now let us find a bound for $||\mathbf{b}||$.

$$\mathbf{b} = \sum_{\substack{k=1\\k\neq i}}^{h} q_{ik}(W_i^\top W_i)(W_i^\top A_i^*)A_k^*$$

$$= \langle W_i, W_i\rangle\langle W_i, A_i^*\rangle q_{ik} A_{-i}^* \mathbf{1}_h$$

Where A_{-i}^* is the dictionary A^* with the ith column set to zero, and $\mathbf{1}_h \in \mathbb{R}^h$ is the h-dimensional vector of all ones. Here we make use of the distributional assumption that q_{ik} is the same for all i, k in order to pull q_{ik} out of the sum.

$$||\mathbf{b}||_2 = h^{2p-2}\langle W_i, W_i\rangle\langle W_i, A_i^*\rangle||A_{-i}^*\mathbf{1}_h||_2$$

$$\leq h^{2p-2}(1+\delta)^3||A_{-i}^*||_2||\mathbf{1}_h||_2$$

$$= h^{2p-2}(1+\delta)^3 h^{1/2}\sqrt{\lambda_{\max}(A_{-i}^{*\top}A_{-i}^*)}$$

$$= h^{2p-2}(1+\delta)^3 h^{1/2}\sqrt{h\frac{\mu}{\sqrt{n}}+1}$$

$$= h^{p-1}\sqrt{h^{2p-2}\times h\times(1+\delta)^6\times\left(h\frac{\mu}{\sqrt{n}}+1\right)}$$

$$= h^{p-1}\sqrt{h^{2p-1}\times(1+h^{-p-\nu^2})^6\times(h^{1-\xi}+1)}$$

$$= h^{p-1}\sqrt{(1+h^{-p-\nu^2})^6\times(h^{2p-\xi}+h^{2p-1})}$$

Here $||A_{-i}^*||_2$ is the spectral norm of A_{-i}^*, and is the top singular value of the matrix. We use Gershgorin's Circle theorem to bound the top eigenvalue of $A_{-i}^{*\top}A_{-i}^*$ by its maximum row sum.

If $p < \frac{\xi}{2}$, $p < \frac{1}{2}$, and $p < \nu^2$, then $||\hat{e_{i4}}|| = o(m_1^2 h^{p-1})$. Now we combine the above obtained bounds for $\|\hat{e_{it}}\|$ (for $t \in \{1,2,3,4\}$) with the bound obtained below equation 2.9 to say that, $\|e_i\| = o(\max\{m_1^2, m_2\}h^{p-1})$

2.9.4 About $\alpha_i - \beta_i$

Remembering that $D = 1$ and doing a close scrutiny of the terms in 2.8 and 2.5 will indicate that the coefficients are the *same* for the $m_2 h^{p-1}$ term in each of them. (which is the term with the highest h scaling in the m_2 dependent parts of α_i and β_i). So this largest term cancels off in the difference and we are left with the sub-leading order terms coming from both their m_1^2 as well as the m_2 parts and this gives us,

$$\alpha_i - \beta_i = o(\max\{m_1^2, m_2\}h^{p-1})$$

References

Rangamani, Akshay, Anirbit Mukherjee, Amitabh Basu, Ashish Arora, Tejaswini Ganapathi, Sang Chin, and Trac D Tran (2018). “Sparse coding and autoencoders”. In: *2018 IEEE International Symposium on Information Theory (ISIT)*. IEEE, pp. 36–40.

Tillmann, Andreas M (2015). “On the computational intractability of exact and approximate dictionary learning”. In: *IEEE Signal Processing Letters* 22.1, pp. 45–49.

Gilbert, Anna (2017). “CBMS Conference on Sparse Approximation and Signal Recovery Algorithms, May 22-26, 2017 and 16th New Mexico Analysis Seminar, May 21”. In: *https://www.math.nmsu.edu/ jlakey/cbms2017/cbms_lecture_notes.html*.

Schnass, Karin (2015a). “A personal introduction to theoretical dictionary learning”. In: *Internationale Mathematische Nachrichten* 228, pp. 5–15.

Olshausen, Bruno A and David J Field (1997). “Sparse coding with an overcomplete basis set: A strategy employed by V1?” In: *Vision research* 37.23, pp. 3311–3325.

Olshausen, Bruno A and David J Field (2005). “How close are we to understanding V1?” In: *Neural computation* 17.8, pp. 1665–1699.

Donoho, David L and Xiaoming Huo (2001). “Uncertainty principles and ideal atomic decomposition”. In: *IEEE Transactions on Information Theory* 47.7, pp. 2845–2862.

Aharon, Michal, Michael Elad, and Alfred Bruckstein (2006). “*k*-SVD: An algorithm for designing overcomplete dictionaries for sparse representation”. In: *IEEE Transactions on signal processing* 54.11, pp. 4311–4322.

Spielman, Daniel A, Huan Wang, and John Wright (2012). “Exact Recovery of Sparsely-Used Dictionaries.” In: *COLT*, pp. 37–1.

Błasiok, Jarosław and Jelani Nelson (2016). “An improved analysis of the ER-SpUD dictionary learning algorithm”. In: *arXiv:1602.05719*.

Agarwal, Alekh, Animashree Anandkumar, Prateek Jain, Praneeth Netrapalli, and Rashish Tandon (2014). "Learning Sparsely Used Overcomplete Dictionaries." In: *COLT*, pp. 123–137.

Arora, Sanjeev, Rong Ge, Tengyu Ma, and Ankur Moitra (2015). "Simple, efficient, and neural algorithms for sparse coding." In: *COLT*, pp. 113–149.

Sun, Ju, Qing Qu, and John Wright (2015). "Complete Dictionary Recovery over the Sphere I: Overview and the Geometric Picture". In: *CoRR* abs/1511.03607. arXiv: `1511.03607`. URL: `http://arxiv.org/abs/1511.03607`.

Barak, Boaz, Jonathan A. Kelner, and David Steurer (2014). "Dictionary Learning and Tensor Decomposition via the Sum-of-Squares Method". In: *CoRR* abs/1407.1543. arXiv: `1407.1543`. URL: `http://arxiv.org/abs/1407.1543`.

Remi, Rémi and Karin Schnass (2010). "Dictionary IdentificationâĂŤSparse Matrix-Factorization via l1 Minimization". In: *IEEE Transactions on Information Theory* 56.7, pp. 3523–3539.

Geng, Quan and John Wright (2014). "On the local correctness of l1-minimization for dictionary learning". In: *Information Theory (ISIT), 2014 IEEE International Symposium on*. IEEE, pp. 3180–3184.

Schnass, Karin (2015b). "Local identification of overcomplete dictionaries." In: *Journal of Machine Learning Research* 16, pp. 1211–1242.

Makhzani, Alireza and Brendan Frey (2013). "K-sparse autoencoders". In: *arXiv preprint arXiv:1312.5663*.

Makhzani, Alireza and Brendan J Frey (2015). "Winner-take-all autoencoders". In: *Advances in Neural Information Processing Systems*, pp. 2791–2799.

Olshausen, Bruno A and David J Field (1996). "Emergence of simple-cell receptive field properties by learning a sparse code for natural images". In: *Nature* 381.6583, p. 607.

Li, Yuanzhi and Yang Yuan (2017). "Convergence Analysis of Two-layer Neural Networks with ReLU Activation". In: *arXiv preprint arXiv:1705.09886*.

Tian, Yuandong (2017). "An Analytical Formula of Population Gradient for two-layered ReLU network and its Applications in Convergence and Critical Point Analysis". In: *arXiv preprint arXiv:1703.00560*.

Chapter 3

A Scale Invariant Measure of Flatness for Deep Network Minima

3.1 Introduction

In the past few years, while deep learning (LeCun, Bengio, and Hinton, 2015) has had empirical successes in several domains such as object detection and recognition (Krizhevsky, Sutskever, and Hinton, 2012; Ren et al., 2015), machine translation (Sutskever, Vinyals, and Le, 2014; Jean et al., 2014), and speech recognition (Hinton et al., 2012; Sainath et al., 2013), there is still a gap between their practical performance and our understanding of generalization in deep learning. Several empirical studies (Chaudhari et al., 2016; Keskar et al., 2016) observe that the generalization ability of a deep network model is related to the spectrum of the Hessian matrix of the training loss at the solution obtained during training. It is also noted that solutions with smaller Hessian spectral norm tend to generalize better. These are popularly known as *Flat Minima*, which have been studied since 1995 (Hochreiter and Schmidhuber, 1995; Hochreiter and Schmidhuber, 1997).

The flat minima heuristic is also related to a more formal framework for generalization – PAC-Bayesian analysis of generalization behavior of deep networks. PAC-Bayes bounds (Dziugaite and Roy, 2017) are concerned with analyzing the behavior of solutions drawn from a posterior distribution. One posterior distribution the bounds are valid for are perturbations about the original solution obtained from empirical risk minimization. Neyshabur et al., 2017 relate the generalization of this distribution to the flatness of the minima obtained.

A number of quantitative definitions of flatness have been proposed both recently (Chaudhari et al., 2016; Keskar et al., 2016) as well as in the early literature (Hochreiter and Schmidhuber, 1997). These authors formalize the notions of "flat" or "wide" minima by either measuring the size of the connected region that is within ϵ of the value of the loss function at the minimum or by finding the difference between the maximum value of the loss function and the minimum value within an ϵ-radius ball of the minimum. Note that the second notion of flatness is closely related to the spectral norm of the Hessian of the loss function at the minimum (through a Taylor expansion).

Definition 1 *If $B_2(\epsilon, \theta)$ is the Euclidean ball of radius ϵ centered at a local minimum θ of a loss function L, then the ϵ-sharpness of the minimum is defined as:*

$$\frac{max_{\theta' \in B_2(\epsilon,\theta)} L(\theta') - L(\theta)}{1 + L(\theta)}.$$

However, Dinh et al., 2017 show that deep networks with positively homogeneous layer activations ($\phi(x)$ is positively homogenous if $\phi(\alpha x) = \alpha\phi(x), \forall\alpha > 0$, like the common ReLU activation $\phi_{\text{rect}}(x) = \max(0, x)$) can be rescaled to make their ϵ-sharpness arbitrarily small or large with a simple transformation that implements the same neural network function but have widely different sharpness measures. To formalize this we consider a 2-layer neural network with parameters $\theta = (\theta_1, \theta_2)$ where the network is given by $y = \theta_2 \phi_{\text{rect}}(\theta_1 x)$. We can transform the parameters of the network by $\alpha > 0$ in the following manner: $T_\alpha(\theta) = (\alpha\theta_1, \alpha^{-1}\theta_2)$. We notice that for positively homogeneous activations, the networks parameterized by θ and $T_\alpha(\theta)$ implement the same function.

Theorem 3.1.1 *(Theorem 4 in Dinh et al., 2017) For a one hidden layer rectified neural network of the form $y = \theta_2 \phi_{rect}(\theta_1 x)$ where $\theta = (\theta_1, \theta_2)$ is a minimum for L such that $\nabla^2 L(\theta) \neq 0$, for any real number $M > 0$, we can find a number $\alpha > 0$ such that $||\nabla^2 L(T_\alpha(\theta))||_2 \geq M$.*

In addition to ϵ-sharpness, there are other notions of sharpness like *expected sharpness*. Expected sharpness arises from the PAC-Bayesian framework which provides guarantees on the expected error of a randomized predictor drawn from a "posterior" distribution that depends on the training data. If f_θ is any predictor, we consider a distribution over predictors with weights of the form $\theta + \nu$, where θ is learned from the training set, and ν is a random variable. This "posterior" distribution and its KL distance from a "prior" distribution can be used to bound the generalization performance of the predictor f_θ, and one of the terms in the bound is the expected sharpness:

$$\mathbb{E}_\nu \left[L(\theta + \nu) - L(\theta) \right]$$

This notion of flatness is also related to the trace norm of the Hessian of the loss function at the minimum (through a Taylor expansion).

This tells us that Hessian based measures like ϵ-sharpness and *expected sharpness* are not very meaningful since we can transform the parameters of the network to get as large or small a quantity as we want. This is also the case for other generalization metrics which are related to the Hessian, such as the one proposed by Wang et al., 2018. In this chapter, we propose an alternative measure for quantifying the sharpness/flatness of minima of empirical loss functions. This measure is based on defining a quotient manifold of parameters which gives us a flatness measure that is invariant to rescalings of the form described above. We use our flatness measure to then test whether flatter minima indeed generalize better. In order to obtain minima that have different generalization properties, we use minibatch SGD training with different batch sizes.

The rest of this chapter is organized as follows. In section 3.2 we give a brief overview of manifold geometry and quotient manifolds. In 3.3 we formalize the rescaling that can change the flatness of minima without changing the function and show that the relation described by rescaling is indeed an equivalence relation, which in turn induces a manifold structure in the space of deep network parameters. In section 3.4 we describe an algorithm analogous to the power method that can be used to estimate the spectral norm of the Riemannian Hessian, which in turn can be employed as a measure of flatness of the deep network minima. In section 3.5 we present several experimental results of applying our measure to small-batch vs large-batch training of various deep networks. Our results confirm that minima that generalize better are flatter.

3.1.1 Related Work

In this chapter, we propose a Hessian based measure for the flatness of minima, which follows pioneering works in Hochreiter and Schmidhuber, 1997 and Keskar et al., 2016 in attempting to measure the sharpness/flatness of deep network minima. Flatter minima are believed to be robust to perturbation of the neural network parameters. Novak et al., 2018 connect generalization to the sensitivty of the network to perturbations to the inputs. In a recent work, Wang et al., 2018 obtain a measure of generalization that is also related to the Hessian at the minima, but still have not resolved the rescaling issue that results in arbitrarily large or small Hessian spectra for the same neural network function.

Manifold approaches to training neural networks have mostly focused on batch normalization (Cho and Lee, 2017; Hoffer et al., 2018). A common approach is to restrict the weights of the linear layers to the manifold of weight matrices with unit norm, or an oblique manifold (Huang et al., 2017), or the Stiefel manifold. To the best of our knowledge, we are the first to propose a quotient manifold of neural network parameters and successfully employ it to resolve the question of how to accurately measure the Hessian of the loss function at minima.

3.2 Preliminaries

In this section we recap some basic ideas in differential geometry and quotient manifolds at a high level. For a more rigorous and detailed exposition please refer to Absil, Mahony, and Sepulchre, 2009

3.2.1 Manifolds, Tangent Spaces, Riemannian Metrics, Connections

A *differentiable manifold* $\mathcal{M}$ is a set that comes along with a collection of diffeomorphic coordinate maps or "charts" that maps subsets of $\mathcal{M}$ to subsets in $\mathbb{R}^d$ (where d is the manifold dimension). Each point $x \in \mathcal{M}$ has a *tangent space* $\mathcal{T}_x\mathcal{M}$ which is a d-dimensional vector space which helps us approximate the first order local structure of $\mathcal{M}$. A *Riemannian metric* g gives each tangent space $\mathcal{T}_x\mathcal{M}$ an inner product. If $\eta_x, \xi_x \in \mathcal{T}_x\mathcal{M}$ are two tangent vectors, $g_x(\eta_x, \xi_x)$ is their inner product. Once we define a metric, $(\mathcal{M}, g)$ is called a *Riemannian manifold*.

We can define smooth functions $f : \mathcal{M} \to \mathbb{R}$ and think of tangent vectors at $x \in \mathcal{M}$ as differential operators on functions defined on the manifold ($\eta_x f$ is the directional derivative of f along the tangent vector η_x). The *Riemannian gradient* of a function is the tangent vector whose inner product with every vector in the tangent space yields the directional derivative along that vector. *Retractions* are maps from the tangent space back to the manifold and can be thought of as finding the point on the manifold which is at unit distance from the current point along the specified tangent vector. For example, in a Euclidean space, $\mathcal{E}$, $R_x(\xi) = x + \xi$ is a retraction.

An *affine connection* ∇ on $\mathcal{M}$ generalizes the notion of directional derivative of a vector field to vector fields on $\mathcal{M}$ ($\nabla_\eta \xi$ is analagous to the directional derivative of the vector field ξ along the direction η). Every Riemannian manifold has a unique *Riemannian connection* that is compatible with the metric and is symmetric (Theorem 5.3.1 of Absil, Mahony, and Sepulchre, 2009).

3.2.2 Quotient Manifolds

If a manifold $\overline{\mathcal{M}}$ comes equipped with an equivalence relation $\sim$, then $[x] = \{y \in \overline{\mathcal{M}}, y \sim x\}$ is the equivalence class of x. The set of all equivalence classes $\overline{\mathcal{M}}/\sim = \{[x], x \in \overline{\mathcal{M}}\}$ is called the quotient of $\overline{\mathcal{M}}$ by $\sim$. Under some conditions, $\overline{\mathcal{M}}/\sim$ admits a manifold structure and can be referred to as a quotient manifold ($\mathcal{M} = \overline{\mathcal{M}}/\sim$). Quotient manifolds are usually referred to in the abstract, and quantities on a quotient manifold ($\overline{\mathcal{M}}/\sim$) are usually represented using quantities from the structure manifold or *total manifold* $\overline{\mathcal{M}}$.

Tangent vectors for quotient manifolds are represented using tangent vectors in the total manifold. However, there are infinitely many elements of the tangent space of the total manifold that can represent the tangent vector of a quotient manifold. To handle this, we partition the total tangent space into two subspaces. If $\overline{\mathcal{M}}/\sim$ has a quotient manifold structure, then the equivalence class $[x]$ of $x \in \overline{\mathcal{M}}$ is a submanifold of $\overline{\mathcal{M}}$. We call its tangent space the *vertical space* $\mathcal{V}_x = \mathcal{T}_x([x])$ and the complementary space as the *horizontal space* $\mathcal{H}_x$, $\mathcal{H}_x \bigoplus \mathcal{V}_x = \mathcal{T}_x\overline{\mathcal{M}}$. In some cases, $\mathcal{H}_x$ and $\mathcal{V}_x$ are defined as orthogonal complements of each other. There is a unique vector in $\mathcal{H}_x$ that can represent the corresponding tangent vector in the quotient manifold $\mathcal{M}$, which is called the *horizontal lift* of the tangent vector. Horizontal lifts are always used to represent tangent vectors on quotient manifolds. Functions defined on quotient manifolds are invariant within an equivalence class, and the Riemannian gradient of a function defined on the quotient manifold is an element of the horizontal space.

Let $(\overline{\mathcal{M}}, \overline{g})$ be a Riemannian manifold, and $\overline{\eta}_x, \overline{\xi}_x$ be horizontal lifts of two tangent vectors in the tangent space of the corresponding point on the quotient manifold $\mathcal{M}$. If $\overline{g}_x(\overline{\eta}_x, \overline{\xi}_x) = \overline{g}_y(\overline{\eta}_y, \overline{\xi}_y)$ for $y \sim x$, then we can define a Riemannian metric for $\mathcal{M}$ through the metric $\overline{g}$ for $\overline{\mathcal{M}}$. Similarly Riemannian connections on $\mathcal{M}$ can be defined through Riemannian connections on $\overline{\mathcal{M}}$.

3.3 Characterizing a Quotient Manifold of Deep Network Parameters

Let us define a neural network as a function $F : \mathbb{R}^{n_0} \to \mathbb{R}^{n_L}$ which takes an n_0-dimensional input and outputs an n_L-dimensional vector which could be a vector of class labels or a continuous measurement, depending on the task. We consider neural networks which consist of a series of nonlinear transformations, represented as

$$F_W(\mathrm{x}) = W_L \phi_{L-1}(W_{L-1}\phi_{L-2}(\dots \phi_1(W_1 \mathrm{x}))).$$

Here $W_i \in \mathbb{R}^{n_i \times n_{i-1}}$ is a linear transformation, and ϕ_i is a positively homogeneous nonlinear function, usually applied pointwise to a vector. Each combination of a linear and nonlinear transformation is referred to as a "layer", and the linear transformation W_i is referred to as the parameter or weights of the layer. Even if W_i has a matrix/convolutional structure, we will be concerned only with the vectorized version, $\mathrm{vec}(W_i) \in \mathbb{R}^{d_i}$, which we will use interchangeably with W_i. First, we consider networks without bias vectors in each layer. We will extend our manifold construction to networks with bias at the end of this section. The proofs of Propositions 3.3.1 and 3.3.2 are given in section 3.7.

Proposition 3.3.1 *Let $W = (W_1, \ldots, W_L) \in \mathbb{R}_*^{d_1} \times \ldots \times \mathbb{R}_*^{d_L}$ be the parameters of a neural network with L layers, and $\lambda = (\lambda_1, \ldots, \lambda_L) \in \mathbb{R}_+^L$ be a set of multipliers. Here $\mathbb{R}_*^{d_i} = \mathbb{R}^{d_i} \backslash \{0\}$. We can transform the layer weights by λ in the following manner: $T_\lambda(W) = (\lambda_1 W_1, \ldots, \lambda_L W_L)$. We introduce a relation from $\mathbb{R}_*^{d_1} \times \ldots \times \mathbb{R}_*^{d_L}$ to itself, $W \sim Y$ if $\exists \lambda$ such that $Y = T_\lambda(W)$ and $\prod_{i=1}^{L} \lambda_i = 1$. Then, the relation $\sim$ is an equivalence relation.*

This equivalence relation is of interest to us because if $W \sim Y$, $F_W(\mathrm{x}) = F_Y(\mathrm{x})$ for all inputs $\mathrm{x} \in \mathbb{R}^{n_0}$. Denote $\overline{\mathcal{M}}_i = \mathbb{R}^{d_i}$ as the Euclidean vector space and the product manifold $\overline{\mathcal{M}} = \overline{\mathcal{M}}_1 \times \ldots \times \overline{\mathcal{M}}_L$ that covers the entire parameter space. We can use the equivalence relation defined in Proposition 3.3.1 to obtain a quotient manifold induced by the equivalence relation $\mathcal{M} := \overline{\mathcal{M}} / \sim$.

Proposition 3.3.2 *The set $\mathcal{M} := \overline{\mathcal{M}} / \sim$ obtained by mapping all points within an equivalence class to a single point in the set has a quotient manifold structure, making $\mathcal{M}$ a differentiable quotient manifold.*

In order to impart a Riemannian structure to our quotient manifold, we need to define a metric on $\overline{\mathcal{M}}$ that is invariant within an equivalence class.

Proposition 3.3.3 *Let $\overline{\eta}_W$ and $\overline{\xi}_W$ be two tangent vectors at a point $W \in \overline{\mathcal{M}}$. The Riemannian metric $\overline{g} : \mathcal{T}_W\overline{\mathcal{M}} \times \mathcal{T}_W\overline{\mathcal{M}}$ defined by:*

$$\overline{g}_W(\overline{\eta}_W, \overline{\xi}_W) = \sum_{i=1}^{L} \frac{\left\langle \overline{\eta}_{W_i}, \overline{\xi}_{W_i} \right\rangle}{||vec(W_i)||_2^2}$$

is invariant within an equivalence class, and hence induces a metric for $\mathcal{M}$, $g_{\pi(W)} = \overline{g}_W$. Here $\langle \cdot, \cdot \rangle$ is the Euclidean inner product and $\overline{\eta}_{W_i}$, $\overline{\xi}_{W_i}$ are the components of $\overline{\eta}_W$, $\overline{\xi}_W$ corresponding to $\overline{\mathcal{M}}_i$.

Proof 3.3.4 *Let U belong to the equivalent class $\pi^{-1}(\pi(W))$, and $\overline{\eta}_W, \overline{\xi}_W$ be tangent vectors in $\mathcal{T}_W\overline{\mathcal{M}}$. Since U and W are in the same equivalence class, $\exists \lambda = (\lambda_1, \ldots, \lambda_L)$ such that $U = (U_1, ..., U_L) = (\lambda_1 W_1, ..., \lambda_L W_L)$ with $\prod_i \lambda_i = 1$. This means that $\overline{\eta}_U = (\overline{\eta}_{U_1}, ..., \overline{\eta}_{U_L}) = (\lambda_1 \overline{\eta}_{W_1}, ..., \lambda_L \overline{\eta}_{W_L})$ (Example 3.5.4 in Absil, Mahony, and Sepulchre, 2009). The same holds for $\overline{\xi}_U$*

Thus,

$$\overline{g}_U(\overline{\eta}_U, \overline{\xi}_U) = \sum_{i=1}^{L} \frac{\left\langle \lambda_i \overline{\eta}_{W_i}, \lambda_i \overline{\xi}_{W_i} \right\rangle}{||\lambda_i vec(W_i)||_2^2} = \overline{g}_W(\overline{\eta}_W, \overline{\xi}_W),$$

Example 3.3.5 *Consider a simple three layer linear neural network $f : \mathbb{R} \to \mathbb{R}$, $f(x) = w\mathbf{u}^\top \mathbf{v}x$ (where $\mathbf{u}, \mathbf{v} \in \mathbb{R}^d$) that is trained using a squared error loss $\ell(w, \mathbf{u}, \mathbf{v}) = \frac{1}{2}(y - w\mathbf{u}^\top \mathbf{v}x)^2$. We can explicity compute the trace of the Riemannian Hessian and see that it is scale invariant. We know that the Riemannian gradient of a function can be computed by hand using the formula $grad\ell(w, \mathbf{u}, \mathbf{v}) = G^{-1}_{(w,\mathbf{u},\mathbf{v})} EGrad\ell(w, \mathbf{u}, \mathbf{v})$ where EGrad refers to the euclidean gradient (Using Eq 3.32 in Absil, Mahony, and Sepulchre, 2009).*

Using the metric defined in 3.3.3, we have that:

$$G_{(w,\mathbf{u},\mathbf{v})} = \begin{bmatrix} \frac{1}{w^2} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \frac{1}{||\mathbf{u}||^2}\mathbf{I}_d & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \frac{1}{||\mathbf{v}||^2}\mathbf{I}_d \end{bmatrix}$$

This means that the trace of the Riemannian Hessian can be computed as:

$$\begin{aligned} Tr(Hess\ \ell(w,\mathbf{u},\mathbf{v})) &= w^2\frac{\partial^2\ell}{\partial w^2} + ||\mathbf{u}||^2 Tr\left(\frac{\partial^2\ell}{\partial \mathbf{u}^2}\right) + ||\mathbf{v}||^2 Tr\left(\frac{\partial^2\ell}{\partial \mathbf{v}^2}\right) \\ &= w^2 \times (\mathbf{u}^\top \mathbf{v} x)^2 + ||\mathbf{u}||^2 \times Tr\left(w^2x^2\mathbf{v}\mathbf{v}^\top\right) + ||\mathbf{v}||^2 \times Tr\left(w^2x^2\mathbf{u}\mathbf{u}^\top\right) \end{aligned}$$

If we compute the Trace of the Riemannian Hessian at a transformed point, that still implements the same function, i.e., $(\lambda_w w, \lambda_\mathbf{u}\mathbf{u}, \lambda_\mathbf{v}\mathbf{v})$ *instead of* $(w,\mathbf{u},\mathbf{v})$ *where* $\lambda_w \times \lambda_\mathbf{u} \times \lambda_\mathbf{v} = 1$, *then we get:*

$$\begin{aligned} Tr(Hess\ \ell(\lambda_w w, \lambda_\mathbf{u}\mathbf{u}, \lambda_\mathbf{v}\mathbf{v})) &= \lambda_w^2 w^2 \times (\lambda_\mathbf{u}\mathbf{u}^\top \lambda_\mathbf{v}\mathbf{v}x)^2 + ||\lambda_\mathbf{u}\mathbf{u}||^2 \times Tr\left(\lambda_w^2 w^2 x^2 \lambda_\mathbf{v}\mathbf{v}\lambda_\mathbf{v}\mathbf{v}^\top\right) \\ &+ ||\lambda_\mathbf{v}\mathbf{v}||^2 \times Tr\left(\lambda_w^2 w^2 x^2 \lambda_\mathbf{u}\mathbf{u}\lambda_\mathbf{u}\mathbf{u}^\top\right) \\ &= (\lambda_w\lambda_\mathbf{u}\lambda_\mathbf{v})^2 \times \Bigg(w^2(\mathbf{u}^\top\mathbf{v}x)^2 + ||\mathbf{u}||^2 Tr\left(w^2x^2\mathbf{v}\mathbf{v}^\top\right) \\ &+ ||\mathbf{v}||^2 Tr\left(w^2x^2\mathbf{u}\mathbf{u}^\top\right) \Bigg) \\ &= Tr(Hess\ \ell(w,\mathbf{u},\mathbf{v})) \end{aligned}$$

3.3.1 Deep Networks with Biases

A deep neural network with biases is a function $F : \mathbb{R}^{n_0} \to \mathbb{R}^{n_L}$ which takes an n_0-dimensional input and outputs an n_L-dimensional vector through a series of nonlinear transformations can be represented as

$$F_{(W,b)}(\mathrm{x}) = W_L \phi_{L-1}(W_{L-1}\phi_{L-2}(W_{L-2} \dots \phi_1(W_1 \mathrm{x} + b_1) \\ \dots + b_{L-2}) + b_{L-1}) + b_L.$$

Here, $b_i \in \mathbb{R}^{n_i}$ are the bias parameters for each layer. Once again, due to the positive homogeneity of the nonlinear functions ϕ_i, we can rescale the weights and biases of the network to obtain a different set of weights and biases that implement the same function.

Suppose we have $\lambda_i \in \mathbb{R}_+, i = 1, \dots, L$, such that $\prod_{i=1}^{L} \lambda_i = 1$. Consider the following transformation:

$$T_\lambda((W, b)) = (\lambda_L W_L, \dots, \lambda_1 W_1, \\ \prod_{i=1}^{L} \lambda_i b_L, \prod_{i=1}^{L-1} \lambda_i b_{L-1}, \dots, \lambda_1 b_1).$$

Now, if $(Y, c) = T_\lambda((W, b))$, then $F_{(W,b)}(\mathrm{x}) = F_{(Y,c)}(\mathrm{x})$ for all $\mathrm{x} \in \mathbb{R}^{n_0}$. Let us denote $\overline{\mathcal{M}}_i = \mathbb{R}^{d_i} \times \mathbb{R}^{n_i}$, as the Euclidean space for each layer. The product space $\overline{\mathcal{M}} = \overline{\mathcal{M}}_1 \times \dots \times \overline{\mathcal{M}}_L$ is the entire space of parameters for the neural networks with biases. Using arguments similar to Propositions 3.3.1 and 3.3.2, we can see that this new transformation also introduces an equivalence relation on $\overline{\mathcal{M}}$ and that $\mathcal{M} := \overline{\mathcal{M}}/\sim$ admits a quotient manifold structure. We provide the formal propositions and their proofs in section 3.7. We modify

Proposition 3.3.3 slightly to get a new metric for the tangent space of $\overline{\mathcal{M}}$.

Proposition 3.3.6 *Let $\overline{\eta}_{(W,b)}$ and $\overline{\xi}_{(W,b)}$ be two tangent vectors at a point $(W,b) \in \overline{\mathcal{M}}$. The Riemannian metric $\overline{g} : \mathcal{T}_{(W,b)}\overline{\mathcal{M}} \times \mathcal{T}_{(W,b)}\overline{\mathcal{M}}$ defined by:*

$$\overline{g}_{(W,b)}(\overline{\eta}_{(W,b)}, \overline{\xi}_{(W,b)}) = \sum_{i=1}^{L} \left(\frac{\left\langle \overline{\eta}_{W_i}, \overline{\xi}_{W_i} \right\rangle}{||vec(W_i)||_2^2} + \frac{\left\langle \overline{\eta}_{b_i}, \overline{\xi}_{b_i} \right\rangle}{||b_i||_2^2} \right)$$

is invariant within an equivalence class and hence induces a metric for $\mathcal{M}$, $g_{\pi(W,b)} = \overline{g}_{(W,b)}$. Here, $\langle \cdot, \cdot \rangle$ is the usual Euclidean inner product and $(\overline{\eta}_{W_i}, \overline{\eta}_{b_i})$ and $(\overline{\xi}_{W_i}, \overline{\xi}_{b_i})$ are are the components of $\overline{\eta}_{(W,b)}$, $\overline{\xi}_{(W,b)}$ corresponding to $\overline{\mathcal{M}}_i$.

3.4 Computing the Scale Invariant Flatness Measure

In the previous section, we introduced a quotient manifold structure that captures the rescaling that is natural to the space of parameters of neural networks with positively homogeneous activations. Now, similar to how the spectral norm of the Euclidean Hessian is used as a measure of flatness, we can use the Taylor expansion of real-valued functions on a manifold to give us an analogous measure of flatness using the spectral norm of the Riemannian Hessian. The definition of Riemannian Hessian as per Absil, Mahony, and Sepulchre, 2009 is as follows.

Definition 2 *For a real valued function f on a Riemannian manifold $\mathcal{M}$, the* Riemannian Hessian *is the linear mapping of $\mathcal{T}_x\mathcal{M}$ onto itself, defined by*

$$Hess f(W)[\xi_W] = \nabla_{\xi_W} grad f$$

for all $\xi_W \in \mathcal{T}_W\mathcal{M}$, where ∇ is a Riemannian connection defined on $\mathcal{M}$.

To see how the Riemmannian Hessian is related to the flatness of the function f around a minimum W, we consider a retraction $R_W : \mathcal{T}_W\mathcal{M} \to \mathcal{M}$. The flatness of a function around a minimum is defined (similar to Definition 1) using the value of the function in a "neighborhood" of the minimum. To formalize what we mean by an ϵ-neighborhood of W, it is the set of points that can be reached through a retraction using tangent vectors of norm at most ϵ, $B_2(\epsilon, W) = \{R_W(\xi), ||\xi||_g \leq \epsilon\}$. Here $|| \cdot ||_g$ is the norm induced by the Riemannian metric g. This gives us the following flatness measure:

$$\frac{\max_{W' \in B_2(\epsilon,W)} f(W') - f(W)}{1 + f(W)}.$$

Using the fact that $\mathcal{T}_W\mathcal{M}$ is a vector space, and that $\hat{f}_W = f \circ R_W$ is a function on a vector space that admits a Taylor expansion, we get the following approximation for $f(W')$ when $W' \in B_2(\epsilon, W)$, and $W' = R_W(\xi_W)$:

$$f(W') \approx f(W) + g(\mathrm{grad} f(W), \xi_W) + \frac{1}{2} g(\xi_W, \mathrm{Hess} f(W)[\xi_W])$$

Using the approximation, recognizing that at a minimum, $\mathrm{grad} f(W) = 0$, and using a Cauchy-Schwarz argument, we can bound the flatness measure by the spectral norm of the Riemannian Hessian. We define it similar to the spectral norm of a linear map in Euclidean space.

Definition 3 *The spectral norm of the Riemannian Hessian of a function $f : \mathcal{M} \to$*

$\mathbb{R}$ *is defined as*

$$||Hessf(W)||_{2,g} = \max_{\xi_W \in \mathcal{T}_W\mathcal{M}, ||\xi_W||_g=1} g(\xi_W, Hessf(W)[\xi_W])$$

With the definition of the spectral norm of the Riemannian Hessian, we now would like to be able to compute it for any function defined on a manifold. To achieve this, we present a Riemannian Power Method in Algorithm 1.

Algorithm 1 Riemannian Power Method

```
1: procedure RiemannianPM($f, W$)
2:     Initialize $\xi_W^0$ randomly in $\mathcal{T}_W\mathcal{M}$
3:     while not converged do    (We use relative change in the eigenvector
   as a stopping criterion)
4:         $\xi_W^{t+1/2} \leftarrow \text{Hess}f(W)[\xi_W^t]$
5:         $\xi_W^{t+1} \leftarrow \frac{\xi_W^{t+1/2}}{||\xi_W^{t+1/2}||_g}$
6:         $t \leftarrow t+1$
7:     end while
8:     return $\xi_W^t$
9: end procedure
```

3.4.1 Implementation

Our procedure to compute the flatness measure is presented in Algorithm 1. Similar to the power method for computing the spectral norm of linear maps in Euclidean space each iteration involves finding the map-vector product, which in our case is a Hessian-vector product, followed normalizing the updated iterate of the top eigenvector of the Hessian.

3.4.1.1 Mathematical Expression for Riemannian Hessian-vector product:

Using Proposition 5.5.2 from Absil, Mahony, and Sepulchre, 2009, we have:

$$g(\xi, \text{Hess}f[\xi]) = \xi(\xi f) - (\nabla_\xi \xi)f$$

Since we are only interested in computing the product between the Hessian and the tangent vector at W, ξ_W, let us set ξ such that it evaluates to ξ_W at W and $\left(\nabla_\xi \xi\right)_W = 0$. One possible choice for ξ for our quotient manifold is such that its total manifold representation $\overline{\xi}$ is constant. The Riemannian connection ∇ of a quotient manifold $\mathcal{M} = \overline{\mathcal{M}}/\sim$ is defined through the Riemannian connection $\overline{\nabla}$ of the total manifold $\overline{\mathcal{M}}$ (Prop 5.3.3 in Absil, Mahony, and Sepulchre, 2009). We define $\overline{\nabla}$ by extending the connection in Theorem 3.4 of Absil, Mahony, and Sepulchre, 2004 to the product manifold.

$$\nabla_\xi \xi = \mathcal{P}^{\mathcal{H}}\left(\left(\overline{\nabla}_{\overline{\xi}}\overline{\xi}\right)_W\right) = \mathcal{P}^{\mathcal{H}}\left(\left.\frac{d}{dt}\overline{\xi}_{W+t\overline{\xi}_W}\right|_{t=0}\right) = 0$$

This means that $g(\xi, \text{Hess}f[\xi]) = \xi(\xi f)$. Now from the definition of the Riemannian gradient of a function (equation 3.31 of Absil, Mahony, and Sepulchre, 2009), we have that $\xi f = g(\text{grad}f, \xi)$ and $\xi(\xi f) = g(\xi, \text{Hess}f[\xi]) = g(\xi, \text{grad}\ g(\text{grad}f, \xi))$ Which means the Riemannian Hessian vector product can be computed as:

$$\text{Hess}f(W)[\xi_W] = \text{grad}\ g(\text{grad}f(W), \xi_W)$$

3.4.1.2 Practical Implementation of Riemannian Hessian-vector product:

As we noted in section 3.2, the abstract quotient manifold gradients and Hessian-vector products are represented using their counterparts in the total manifold, which in our case is the space of deep network parameters. We implement the Riemannian Hessian-vector product using the steps below.

1. Find the Euclidean gradient of f (EGrad$f(W)$) using backpropagation
2. Compute the representation of the Riemannian gradient of f as $\overline{\text{grad} f(W)} = \overline{G}_W^{-1}\text{EGrad} f(W)$ (Eq 3.32 in Absil, Mahony, and Sepulchre, 2009). Here $\overline{G}_W^{-1}$ is the inverse of the matrix representation of the metric (g_W) at W, given by $\overline{G}_W^{-1} = \text{diag}(\ldots, ||\text{vec}(W_i)||^2 I_{d_i \times d_i}, \ldots)$
3. Compute the inner product (as defined by our Riemannian metric) between $\overline{\text{grad} f(W)}$ and $\overline{\xi}_W$ (representation of vector whose Hessian-vector product is desired)
4. Find the Riemannian gradient of the inner product by first finding its Euclidean gradient using backpropagation and subsequently premultiplying by $\overline{G}_W^{-1}$

3.4.2 Simulations

To validate our Riemannian Power Method Algorithm, we consider two deep network architectures described in Table 3.1. For each architecture, we generate a synthetic dataset containing $N = 500$ samples in $\mathbb{R}^{784}$ which belong to one of 10 different classes with randomly generated class labels. For each network, we consider softmax cross-entropy as the loss function.

Network	**Architecture**
F_1	[FC(784, 300), FC(300, 100), FC(100, 10)]
C_1	[conv(5, 5, 10), conv(5, 5, 20), FC(320, 120), FC(120, 84), FC(84, 10)]

Table 3.1: Network Architectures for Simulations

We compute the spectral norms of the Hessians of their losses at different points within the equivalence class by considering $(Y, c) = T_\lambda((W, b))$ for different settings of λ. Let $\sigma_{(W,b)}$ be the spectral norm computed at (W, b), and $\sigma_{(Y,c)}$ be the spectral norm computed at (Y, c). We define the relative difference between the two measurements as follows:

$$\texttt{Relative Difference} = \frac{|\sigma_{(W,b)} - \sigma_{(Y,c)}|}{\sigma_{(W,b)}}$$

Results for F_1 are reported in Table 3.2 whereas results for C_1 are reported in Table 3.3.

λ	Relative Difference
$(5, 4, \frac{1}{20})$	1.7×10^{-7}
$(100, 30, \frac{1}{3000})$	7.17×10^{-7}

Table 3.2: `Relative Difference` in Spectral Norms for F_1 under different transformations

λ	Relative Difference
$(5, 4, 3, 2, \frac{1}{120})$	1.28×10^{-7}
$(50, 24, 30, \frac{1}{6}, \frac{1}{6000})$	5.1×10^{-6}

Table 3.3: `Relative Difference` in Spectral Norms for C_1 under different transformations averaged over 20 runs (negligible variance)

In Figure 3.1, we can observe how our power method based algorithm converges for an F_1 network. From the tables, we notice that the spectral norm that we compute using the eigenvectors obtained using the Riemannian Power Method is invariant to transformations within the equivalence class. That is, the values for `Relative Difference` are small.

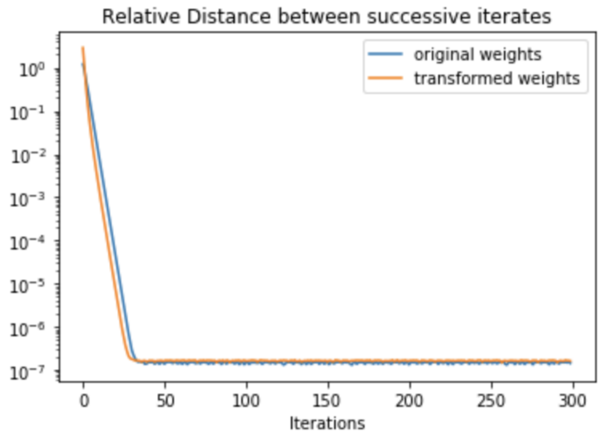


Figure 3.1: Convergence of Riemannian Power Method for a synthetic dataset for an F_1 network averaged over 20 runs (negligible variance)

3.5 Experiments

Now that we have proposed a measure of flatness for deep network minima, we turn to the empirical question at hand - does flatness correlate with the generalization ability of the deep network? We have so far established a quantitative measure of flatness (spectral norm of the Riemannian Hessian) that will allow us to answer this question. Now in order to test this proposition, we need a way to find global minima of the deep network loss which may generalize worse or generalize better.

In order to find these solutions we turn to large-batch training vs small-batch training of neural networks. In an empirical study Keskar et al., 2016 observe that small-batch gradient methods with 32-512 samples per batch tend to converge to flatter minima than large-batch methods which have batch sizes of the order of 1000s of samples. However, since Dinh et al., 2017 have shown that measures of flatness can be gamed by rescaling the network appropriately, we cannot trust the current quantitative measures to compare the flatness of these solutions. Instead, we use the spectral norm of the Riemannian Hessian as a measure of flatness and compare the flatness of the solutions obtained using large-batch and small-batch training. Our goal in this set of experiments is not to achieve state of the art performance on these datasets. Instead, we are interested in characterizing the flatness of the solutions obtained and studying how that correlates with test set accuracy. The datasets and network architectures used in our experiments are listed in Table 3.4.

Dataset	**Network 1**	**Network 2**
MNIST	MNIST-FC 512 fully-connected $\times 5$	LeNet 2 conv-pool layers, 120, 84 fully-connected
Fashion MNIST	MNIST-FC	LeNet
KMNIST	MNIST-FC	LeNet
CIFAR10	AlexNet shallow convnet	VGG16 deep convnet

Table 3.4: Datasets and Deep Network Architectures used in our experiments

Our experiments were run on a machine with 4 Tesla P40 GPUs with 24GB of GPU memory each. The machine itself has 256 GB of RAM and an Intel Xeon processor with 24 cores. Even so, we ran into memory issues when trying to estimate the flatness for AlexNet and VGG16 on CIFAR10. We had to use a subset of the training set to approximate the empirical loss function. We used 5000 samples for AlexNet and 2000 samples for VGG16. The subset was the same for all batch sizes. Table 3.5 has the details for the other datasets.

Dataset	**Train Size**	**Test Size**	**Samples used for Flatness Measurement**
MNIST	50000	10000	50000
KMNIST	60000	10000	60000
Fashion MNIST	60000	10000	60000
CIFAR10	50000	10000	AlexNet - 5000 VGG16 - 2000

Table 3.5: Train Test splits for the datasets used. Last column reports number of training samples used to estimate our Flatness Measure

The hyperparameters and training algorithms used are reported below in Table 3.6 . These were chosen so as to ensure that all networks could be trained to global optimality on the empirical loss on the training set.

3.5.1 Visualizing the Loss Landscape

We generate parametric line plots along different random directions for AlexNet and VGG16. These plots are shown in Figure 3.2. These plots are *layer normalized* Li et al., 2018, which means that the random directions chosen are scaled according to the norms of the layers of the trained networks. More precisely, if the minimum obtained from training AlexNet/VGG is $W = (W_1, \ldots, W_L)$, we generate random direction $V = (V_1, \ldots, V_L)$, and plot the loss along the curve $\hat{W}(t)$ for $t \in [-1, 1]$. Here $\hat{W}(t)$ is given by:

$$\hat{W}(t) = \left(\ldots, W_i + t \times \frac{||\text{vec}(W_i)||_2}{||\text{vec}(V_i)||_2} V_i, \ldots \right).$$

From the plots we see that the large-batch plots are above the small-batch

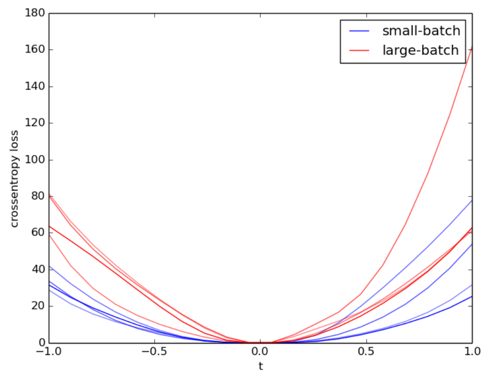
Dataset + Network	Optimizer	Epochs
MNIST-FC	Adam, `lr=1e-3`	small batch - 200 large batch - 500
MNIST-LeNet	Adam, `lr=1e-3`	small batch - 200 large batch - 500
KMNIST-FC	Adam, `lr=1e-3`	small batch - 200 large batch - 500
KMNIST-LeNet	Adam, `lr=1e-3`	small batch - 200 large batch - 500
Fashion MNIST-FC	Adam, `lr=1e-3` 0.5 schedule every 200 epochs	small batch - 500 large batch - 500
Fashion MNIST-LeNet	Adam, `lr=5e-3` 0.5 schedule every 50 epochs	small batch - 200 large batch - 500
CIFAR10 - AlexNet	SGD small batch `lr=1e-3, momentum=0.9` large batch `lr=1e-4, momentum=0.99`	small batch - 200 large batch - 500
CIFAR10 - VGG16	SGD small batch `lr=1e-3, momentum=0.9` large batch `lr=1e-4, momentum=0.99`	small batch - 400 large batch - 600

Table 3.6: Training algorithms and hyperparameters used in our experiments. Last column reports the number of epochs used to train with small batch sizes and large batch sizes respectively.

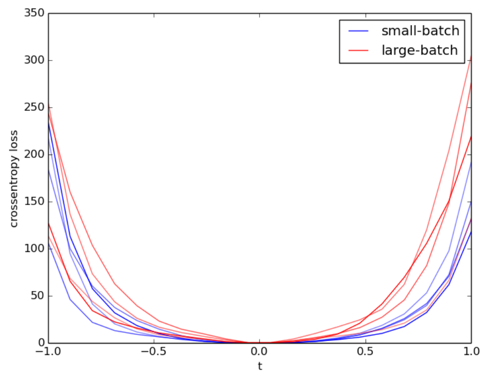
plots, indicating that the large-batch minima are sharper than the small-batch counterparts.

3.5.2 Results

For each network architecture and dataset, we trained the network to 100% training accuracy using SGD or Adam, until the training cross-entropy loss values were smaller than 10^{-6}. This means we attain *global minima* of the cross entropy loss in each case. We would like to note that in our experiments we did not attempt to tune the learning rate as we changed the batch size since our goal was to obtain different global minima with different generalization properties. Adjusting hyperparameters in order to enable large batch training



(a) AlexNet



(b) VGG16

Figure 3.2: Parametric line plots for convolutional networks trained on CIFAR-10

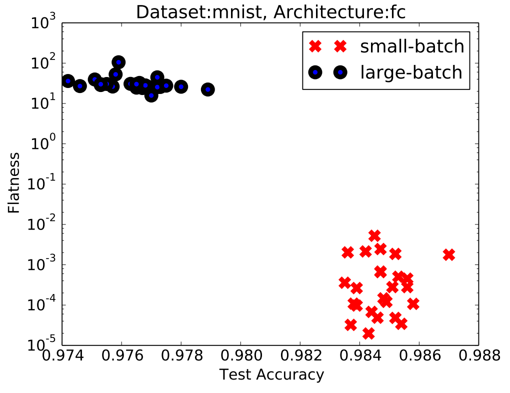
of deep networks is an active area of research, but it is not the focus of our experiments.

Now, in order to quantify the flatness and see how it correlates with generalization, we report the test accuracy and spectral norm of the Riemannian Hessian at minima for each of the networks trained on MNIST and CIFAR10 in Table 3.7. We observe that the estimated spectral norms for the large-batch minima are orders of magnitude larger than those of the small-batch minima for every network and dataset. This also correlates with test accuracy, with the flatter minima having better generalization abilities.

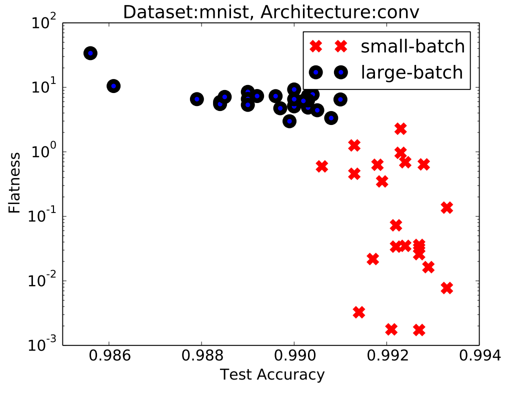
The complete set of results for the experiments we ran are presented in Table 3.7, and all scatter plots of all experimental runs are shown in Figures 3.3, 3.4, 3.5, 3.6. We observe that the estimated spectral norms for the large-batch minima are orders of magnitude larger than those of the small-batch minima for every network and dataset. This also correlates with test accuracy, with the flatter minima having better generalization abilities. In the scatter plots, we can see that the top left and bottom right sections of the plots tend to be populated, meaning minima that are flatter have better generalization performance as measured using the test set than sharper minima.

Batch Size	**Test Accuracy**	**Spectral Norm**
MNIST / Fully-Connected (25 runs)		
200	$98.5 \pm 0.1\%$	0.00086 ± 0.0012
5000	$97.6 \pm 0.1\%$	32.99 ± 17.13
MNIST / LeNet (25 runs)		
200	$99.2 \pm 0.1\%$	0.38 ± 0.55
5000	$98.9 \pm 0.1\%$	7.41 ± 5.64
KMNIST / Fully-Connected (25 runs)		
200	$92.9 \pm 0.1\%$	0.0325 ± 0.0644
5000	$89.7 \pm 0.3\%$	30.45 ± 7.15
KMNIST / LeNet (25 runs)		
200	$95.3 \pm 0.1\%$	0.356 ± 0.531
5000	$93.5 \pm 0.05\%$	2870.35 ± 634.21
Fashion MNIST / Fully-Connected (25 runs)		
200	$90.1 \pm 0.3\%$	7.057 ± 8.83
5000	$89.3 \pm 0.3\%$	7466.98 ± 1494.75
Fashion MNIST / LeNet (25 runs)		
200	$90.8 \pm 0.3\%$	62.17 ± 67.49
5000	$89.5 \pm 0.4\%$	7685.64 ± 4778.15
CIFAR-10 / AlexNet (15 runs)		
200	$72.76 \pm 0.92\%$	12.664 ± 3.589
2000	$67.36 \pm 0.23\%$	406.22 ± 236.49
CIFAR-10 / VGG16 (15 runs)		
200	$75.42 \pm 0.93\%$	19.58 ± 14.97
2000	$65.98 \pm 0.73\%$	300055.02 ± 58257.42

Table 3.7: Test Accuracy and Spectral Norm of Riemannian Hessian at Minima for different trained networks. All quantities reported as Mean $\pm$ Standard Deviation

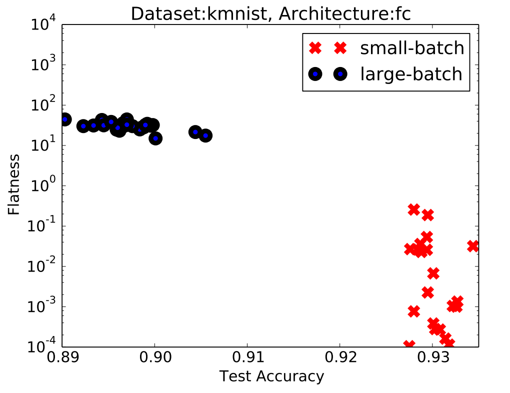


(a) MNIST-FC (25 runs)

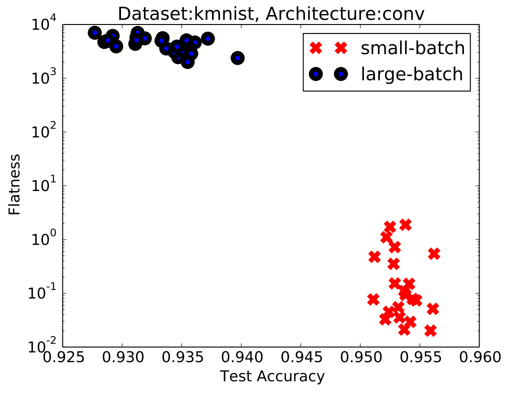


(b) MNIST - LeNet (25 runs)

Figure 3.3: Visualizing the relationship between flatness of minima (as measured by our proposed method) and generalization for MNIST. Smaller flatness measure means the minima is flatter and higher test accuracy means better generalization.

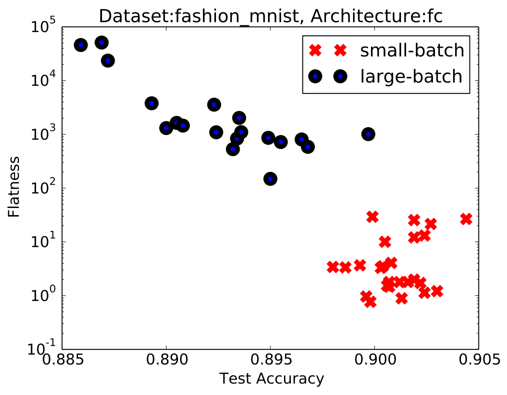


(a) KMNIST-FC (25 runs)

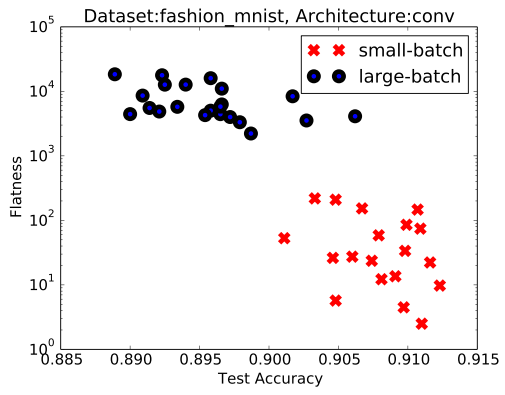


(b) KMNIST - LeNet (25 runs)

Figure 3.4: Visualizing the relationship between flatness of minima (as measured by our proposed method) and generalization for KMNIST. Smaller flatness measure means the minima is flatter and higher test accuracy means better generalization.

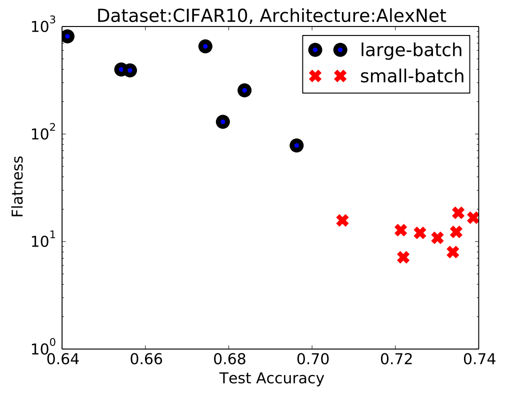


(a) Fashion MNIST-FC (25 runs)

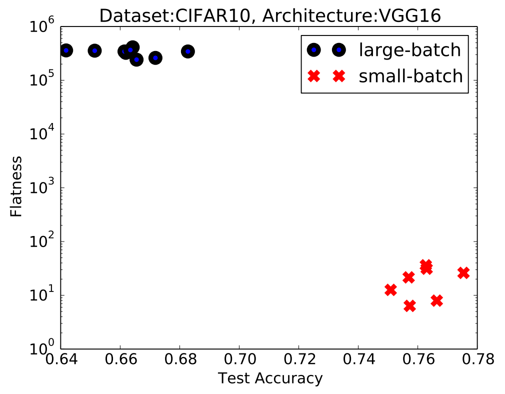


(b) Fashion MNIST - LeNet (25 runs)

Figure 3.5: Visualizing the relationship between flatness of minima (as measured by our proposed method) and generalization for Fashion MNIST. Smaller flatness measure means the minima is flatter and higher test accuracy means better generalization.



(a) CIFAR10 - AlexNet (15 runs)



(b) CIFAR10 - VGG16 (15 runs)

Figure 3.6: Visualizing the relationship between flatness of minima (as measured by our proposed method) and generalization for CIFAR10. Smaller flatness measure means the minima is flatter and higher test accuracy means better generalization.

3.6 Conclusion and Future Work

In this chapter, we observe that natural rescalings of neural networks with positively homogeneous activations induce an equivalence relation in the parameter space which in turn leads to a quotient manifold structure in the parameter space. We provide theoretical justification for these claims and then adopt the manifold structure to propose a Riemannian Hessian based flatness measure for deep network minima. We provide an algorithm to compute this measure and run experiments to confirm that flatter minima tend to generalize better. Our framework provides a principled path to estimate properties of the loss landscape such that they are invariant to rescaling of deep networks.

We believe this quotient manifold view of the parameter space of neural networks can have implications for training deep networks as well. We would like to study how descent techniques on this manifold will compare to algorithms like weight normalization Salimans and Kingma, 2016 and Path-SGD Neyshabur, Salakhutdinov, and Srebro, 2015.

3.7 Missing Proofs from Section 3.3

3.7.1 Deep Networks without Biases

Proposition 3.7.1 *Let $W = (W_1, \ldots, W_L) \in \mathbb{R}_*^{d_1} \times \ldots \times \mathbb{R}_*^{d_L}$ be the parameters of a neural network with L layers, and $\lambda = (\lambda_1, \ldots, \lambda_L) \in \mathbb{R}_+^L$ be a set of multipliers. Here $\mathbb{R}_*^{d_i} = \mathbb{R}^{d_i} \backslash \{0\}$. We can transform the layer weights by λ in the following manner: $T_\lambda(W) = (\lambda_1 W_1, \ldots, \lambda_L W_L)$. We introduce a relation from $\mathbb{R}_*^{n_1 \times n_0} \times \ldots \times \mathbb{R}_*^{n_L \times n_{L-1}}$ to itself, $W \sim Y$ if $\exists \lambda$ such that $Y = T_\lambda(W)$ and $\prod_{i=1}^L \lambda_i = 1$. The* *relation $\sim$, is an equivalence relation.*

Proof 3.7.2 *1. It is self evident that $W \sim W$, with $\lambda = (1, \ldots, 1)$*

2. *If $W \sim Y$, then $\exists \lambda$ such that $Y = T_\lambda(W)$. Set $\tilde{\lambda} = (\lambda_1^{-1}, \ldots, \lambda_L^{-1})$, then $\tilde{\lambda}_i > 0$ and $\prod_{i=1}^{L} \tilde{\lambda}_i = \frac{1}{\prod_{i=1}^{L} \lambda_i} = 1$. Also, $W = T_{\tilde{\lambda}}(Y)$, which means $Y \sim W$.*

3. *Let $W \sim Y$, and $Y \sim Z$. This means, $\exists \lambda^1$ such that $Y = T_{\lambda^1}(W)$, and $\exists \lambda^2$ such that $Z = T_{\lambda^2}(Y)$ Let $\tilde{\lambda} = (\lambda_1^1 \lambda_1^2, \ldots, \lambda_L^1 \lambda_L^2)$. We see that $\tilde{\lambda}_i > 0$, and $\prod_{i=1}^{L} \tilde{\lambda}_i = \prod_{i=1}^{L} \lambda_i^1 \times \prod_{j=1}^{L} \lambda_j^2 = 1$. Since $Z = T_{\tilde{\lambda}}(W)$, we have that $W \sim Z$.*

Hence $\sim$ is an equivalence relation.

Proposition 3.7.3 *The set $\mathcal{M} := \overline{\mathcal{M}} / \sim$ obtained by mapping all points within an equivalence class to a single point in the set has a quotient manifold structure, making $\mathcal{M}$ a differentiable quotient manifold.*

Proof 3.7.4 *In order to prove that $\mathcal{M}$ is a manifold, we need to show that:*

1. *$graph(\sim) = \{(W, Y) : W, Y \in \overline{\mathcal{M}}, W \sim Y\}$ is an embedded submanifold of $\overline{\mathcal{M}} \times \overline{\mathcal{M}}$.*

2. *The projection $\pi_1 : graph(\sim) \to \overline{\mathcal{M}}$, $\pi_1(W, Y) = W$ is a submersion.*

3. *$graph(\sim)$ is a closed subset of $\overline{\mathcal{M}} \times \overline{\mathcal{M}}$.*

First, we look at a point $(W^0, Y^0) \in graph(\sim)$. This means $\exists \lambda \in \mathbb{R}_+^L$, $\prod_{i=1}^{L} \lambda_i = 1$, such that $Y^0 = T_\lambda(W^0)$. For every $V \in \mathbb{R}^{d_1} \times \ldots \times \mathbb{R}^{d_L}$ we can define

$\gamma(t) = (W^0 + tV, T_\lambda(W^0 + tV))$ which is a smooth curve and an injection from $\mathbb{R}$ to graph$(\sim)$, and $\pi_1(\gamma(t)) = W^0 + tV$. Since $\frac{d\pi_1(\gamma(t)}{dt} = V$, we see that dim(range$(D\pi_1)) =$ dim$(\overline{\mathcal{M}})$, where $D\pi_1$ is the Jacobian of π_1. This means that π_1 is a submersion, proving point 2.

Next we will prove point 3. For this, we define a function $F : \overline{\mathcal{M}} \times \overline{\mathcal{M}} \to \mathbb{R}^{d_1} \times \ldots \times \mathbb{R}^{d_L} \times \mathbb{R}$

$$F(W,Y) = \begin{bmatrix} Y_1 - \frac{\langle W_1, Y_1 \rangle}{||vec(W_1)||_2^2} W_1 \\ \vdots \\ Y_L - \frac{\langle W_L, Y_L \rangle}{||vec(W_L)||_2^2} W_L \\ log\left(\frac{\prod_{i=1}^{L} ||vec(Y_i)||_2^2}{\prod_{j=1}^{L} ||vec(W_j)||_2^2} \right) \end{bmatrix}$$

Under F, the preimage of $0_{d_1 \times \ldots \times d_L \times 1}$, is graph$(\sim)$. Since the preimage of a closed set is a closed set, we have that graph$(\sim)$ is a closed subset of $\overline{\mathcal{M}} \times \overline{\mathcal{M}}$.

Finally we will prove 1, by defining a submersion from $\overline{\mathcal{M}}$ to $\mathbb{R}^{d_1 - 1 \times \ldots \times d_L - 1 \times 1}$. Suppose there is a smooth function F_1 from $\overline{\mathcal{M}}$ to $St(d_1 - 1, d_1) \times \ldots \times St(d_L - 1, d_L)$ (where $St(p,n)$ is the p-dimensional Stiefel manifold), such that:

$$F_1(W) = \begin{bmatrix} W_1^\perp \\ \vdots \\ W_L^\perp \end{bmatrix}$$

Here $W_i^\perp$ is an orthogonal basis for the $d_i - 1$ dimensional subspace that is orthogonal to vec(W_i), for all $W \in \overline{\mathcal{M}}$. Such an F_1 always exists, since given W_i we can find $W_i^\perp$ by performing a Gram-Schmidt orthogonalization on $[vec(W_i)|E]$ and taking the last $d_i - 1$ columns. Here E is chosen such that $[vec(W_i)|E]$ is full rank.

Now given F_1, we can define $F_2 : \overline{\mathcal{M}} \times \overline{\mathcal{M}} \to \mathbb{R}^{d_1-1} \times \ldots \times \mathbb{R}^{d_L-1} \times \mathbb{R}$

$$F_2(W,Y) = \begin{bmatrix} (W_1^{\perp})^{\top} vec(Y_1) \\ \vdots \\ (W_L^{\perp})^{\top} vec(Y_L) \\ \sum_{i=1}^{L} log \frac{\langle vec(W_i), vec(Y_i) \rangle}{||vec(W_i)||_2^2} \end{bmatrix}$$

For any $[X_1, \ldots, X_L, x] \in \mathbb{R}^{d_1-1} \times \ldots \times \mathbb{R}^{d_L-1} \times \mathbb{R}$, we can define $\tilde{Y}$ such that

$$vec(\tilde{Y}) = \left[W_1^{\perp} X_1 + \frac{x}{L} vec(Y_1), \ldots, W_L^{\perp} X_L + \frac{x}{L} vec(Y_L) \right]$$

which means, for points $(W, Y) \in graph(\sim)$:

$$\begin{aligned} DF_2(W,Y)[0,\tilde{Y}] = \Bigg[&(W_1^{\perp})^{\top} W_1^{\perp} X_1 + \frac{x}{L} (W_1^{\perp})^{\top} vec(Y_1), \\ &\ldots, (W_L^{\perp})^{\top} W_L^{\perp} X_L + \frac{x}{L} (W_L^{\perp})^{\top} vec(Y_L), \\ &\sum_{i=1}^{L} \frac{||vec(W_i)||_2^2}{\langle vec(W_i), vec(Y_i) \rangle} \\ &\times \frac{vec(W_i)^{\top} \left(W_i^{\perp} X_i + \frac{x}{L} vec(Y_i) \right)}{||vec(W_i)||_2^2} \Bigg] \\ &= [X_1, \ldots, X_L, x] \end{aligned}$$

This means that F_2 is a submersion at each point of $graph(\sim)$, and the set $F_2^{-1}(0) = graph(\sim)$ is an embedded submanifold of $\overline{\mathcal{M}} \times \overline{\mathcal{M}}$. This concludes our proof that $\mathcal{M} = \overline{\mathcal{M}} / \sim$ is a quotient manifold.

3.7.1.1 Characterizing the Vertical Tangent Space of the Quotient Manifold

One invariant property of the equivalence class is the product of the norms of all the layers. That is, if $U \in \approx^{-1}(\approx(W))$, then $\prod_i \|\text{vec}(U_i)\|_2^2 = \prod_i \|\text{vec}(W_i)\|_2^2$. For calculation convenience, we can replace the product by the sum by applying the log operator which gives $\sum_i \log \|\text{vec}(U_i)\|_2^2 = \sum_i \log \|\text{vec}(W_i)\|_2^2$.

Lemma 3.7.5 *The tangent space of $\approx^{-1}(\approx(W))$ at U is $(\beta_1 U_1, ..., \beta_L U_L)$ with $\sum_i \beta_i = 0$.*

Proof 3.7.6 *Consider the curves $U_i(t) \in \overline{\mathcal{M}}_i$ with $U_i(0) = U_i$, we have*

$$\sum_i \log \|vec(U_i(t))\|_2^2 = \sum_i \log \|vec(W_i)\|_2^2 .$$

Taking the derivative on both sides with respect to t gives

$$\sum_i \frac{\langle \dot{U}_i(t), U_i(t) \rangle}{\|vec(U_i(t))\|_2^2} = 0.$$

It is clear that $\dot{U}_i(t) = \beta_i U_i(t)$ with $\sum_i \beta_i = 0$ satisfies the above equation. Therefore the tangent space $\mathcal{T}_U$ of $\approx^{-1}(\approx(W))$ contains all tangent vectors $\dot{U} = (\beta_1 U_1, ..., \beta_L U_L)$ with $\sum_i \beta_i = 0$.

The tangent space to the embedded submanifold $\approx^{-1}(\approx(W))$ of $\overline{\mathcal{M}}$ is usually referred to as the Vertical Tangent space ($\mathcal{V}_W$) of the quotient manifold $\mathcal{M}$. The orthogonal complement of the vertical space from the tangent space $\mathcal{T}_W\overline{\mathcal{M}}$ is referred to as the horizontal space $\mathcal{H}_W$. We note that all smooth curves $\gamma(t) : \mathbb{R} \to \overline{\mathcal{M}}$ such that $\gamma(0) = W$ and $\dot{\gamma}(0) \in \mathcal{V}_W$, lie within the equivalence class $\approx^{-1}(\approx(W))$.

3.7.2 Deep Networks with Biases

We recall that Deep Networks with biases are defined as follows:

$$F_{(W,b)}(\mathrm{x}) = W_L\phi_{L-1}(W_{L-1}\phi_{L-2}(W_{L-2}\dots\phi_1(W_1\mathrm{x}+b_1)$$
$$\dots + b_{L-2}) + b_{L-1}) + b_L$$

The equivalence relation for the parameter space of deep networks with biases is defined through the following transformation. Suppose we have $\lambda_i \in \mathbb{R}_+, i = 1,\dots,L$, such that $\prod_{i=1}^{L}\lambda_i = 1$. Consider the following transformation:

$$T_\lambda((W,b)) = (\lambda_L W_L,\dots,\lambda_1 W_1,$$
$$\prod_{i=1}^{L}\lambda_i b_L, \prod_{i=1}^{L-1}\lambda_i b_{L-1},\dots,\lambda_1 b_1)$$

Now, if $(Y,c) = T_\lambda((W,b))$, then $F_{(W,b)}(\mathrm{x}) = F_{(Y,c)}(\mathrm{x}), \forall \mathrm{x} \in \mathbb{R}^{n_0}$. Thus we define the equivalence relation $\sim$, where $(Y,c) \sim (W,b)$ if $\exists\lambda$ such that $(Y,c) = T_\lambda((W,b))$.

Let us denote $\overline{\mathcal{M}}_i = \mathbb{R}^{d_i} \times \mathbb{R}^{n_i}$, as the Euclidean space for each layer. The product space $\overline{\mathcal{M}} = \overline{\mathcal{M}}_1 \times \dots \times \overline{\mathcal{M}}_L$ is the entire space of parameters for neural networks with biases.

Proposition 3.7.7 *The set $\mathcal{M} := \overline{\mathcal{M}}/\sim$ obtained by mapping all points within an* *equivalence class to a single point in the set has a quotient manifold structure, making $\mathcal{M}$ a differentiable quotient manifold.*

Proof 3.7.8 *In order to prove that $\mathcal{M}$ is a manifold, we need to show that:*

1. *$graph(\sim) = \{((W,b),(Y,c)) : (W,b),(Y,c) \in \overline{\mathcal{M}}, (W,b) \sim (Y,c)\}$ is an embedded submanifold of $\overline{\mathcal{M}} \times \overline{\mathcal{M}}$.*

2. *The projection $\pi_1 : graph(\sim) \to \overline{\mathcal{M}}$, $\pi_1((W,b),(Y,c)) = (W,b)$ is a submersion.*

3. *$graph(\sim)$ is a closed subset of $\overline{\mathcal{M}} \times \overline{\mathcal{M}}$.*

First, we look at a point $((W^0,b^0),(Y^0,c^0)) \in graph(\sim)$. This means $\exists \lambda \in \mathbb{R}^L_+$, $\prod_{i=1}^L \lambda_i = 1$, such that $(Y^0,c^0) = T_\lambda((W^0,b^0))$. For every $(V,v) \in \mathbb{R}^{d_1} \times \ldots \times \mathbb{R}^{d_L} \times \mathbb{R}^{n_1} \times \ldots \times \mathbb{R}^{n_L}$ we can define $\gamma(t) = ((W^0,b^0) + t(V,v), T_\lambda((W^0,b^0) + t(V,v)))$ which is a smooth curve and an injection from $\mathbb{R}$ to $graph(\sim)$, and $\pi_1(\gamma(t)) = (W^0,b^0) + t(V,v)$. Since $\frac{d\pi_1(\gamma(t)}{dt} = (V,v)$, we see that $dim(range(D\pi_1)) = dim(\overline{\mathcal{M}})$, where $D\pi_1$ is the Jacobian of π_1. This means that π_1 is a submersion, proving point 2.

Next we will prove point 3. For this, we define a function $F : \overline{\mathcal{M}} \times \overline{\mathcal{M}} \to \mathbb{R}^{d_1} \times \ldots \times \mathbb{R}^{d_L} \times \mathbb{R}^{n_1} \times \ldots \times \mathbb{R}^{n_L} \times \mathbb{R}^{L+1}$

$$F((W,b),(Y,c)) = \begin{bmatrix} Y_1 - \frac{\langle W_1, Y_1 \rangle}{||vec(W_1)||_2^2} W_1 \\ \vdots \\ Y_L - \frac{\langle W_L, Y_L \rangle}{||vec(W_L)||_2^2} W_L \\ c_1 - \frac{\langle b_1, c_1 \rangle}{||b_1||_2^2} b_1 \\ \vdots \\ c_L - \frac{\langle b_L, c_L \rangle}{||b_L||_2^2} b_L \\ log\left(\frac{\prod_{i=1}^{L} ||vec(Y_i)||_2^2}{\prod_{j=1}^{L} ||vec(W_j)||_2^2}\right) \\ log\left(\frac{||c_1||_2^2}{||b_1||_2^2}\right) - log\left(\frac{||vec(Y_1)||_2^2}{||vec(W_1)||_2^2}\right) \\ log\left(\frac{||c_2||_2^2}{||b_2||_2^2}\right) - log\left(\frac{||c_1||_2^2}{||b_1||_2^2} \times \frac{||vec(Y_2)||_2^2}{||vec(W_2)||_2^2}\right) \\ \vdots \\ log\left(\frac{||c_L||_2^2}{||b_L||_2^2}\right) - log\left(\frac{||c_{L-1}||_2^2}{||b_{L-1}||_2^2} \times \frac{||vec(Y_L)||_2^2}{||vec(W_L)||_2^2}\right) \end{bmatrix}$$

Under F, the preimage of $0_{d_1 \times \ldots \times d_L \times \times n_1 \times \ldots \times n_L \times L+1}$, is graph$(\sim)$. Since the preimage of a closed set is a closed set, we have that graph$(\sim)$ is a closed subset of $\overline{\mathcal{M}} \times \overline{\mathcal{M}}$.

Finally we will prove 1, by defining a submersion from $\overline{\mathcal{M}}$ to $\mathbb{R}^{d_1-1 \times \ldots \times d_L-1 \times n_1-1 \times \ldots \times n_L-1 \times L+1}$. Suppose there is a smooth function F_1 from $\overline{\mathcal{M}}$ to $St(d_1-1, d_1) \times \ldots \times St(d_L - 1, d_L) \times St(n_1-1, n_1) \times \ldots \times St(n_L-1, n_L)$ (where $St(p,n)$ is the p-dimensional Stiefel manifold), such that:

$$F_1((W,b)) = \begin{bmatrix} W_1^{\perp} \\ \vdots \\ W_L^{\perp} \\ b_1^{\perp} \\ \vdots \\ b_L^{\perp} \end{bmatrix}$$

Here $W_i^{\perp}$ is an orthogonal basis for the $d_i - 1$ dimensional subspace that is orthogonal to $vec(W_i)$, and $b_i^{\perp}$ is an orthogonal basis for the $n_i - 1$ dimensional subspace orthogonal to b_i, for all $(W, b) \in \overline{\mathcal{M}}$. Such an F_1 always exists, since given W_i (alternatively b_i) we can find $W_i^{\perp}$ (alternatively $b_i^{\perp}$) by performing a Gram-Schmidt orthogonalization on $[vec(W_i)|E]$ (or $[b_i|E]$) and taking the last $d_i - 1$ (or $n_i - 1$) columns. Here E is chosen such that $[vec(W_i)|E]$ (or $[b_i|E]$) is full rank.

Now given F_1, we can define $F_2 : \overline{\mathcal{M}} \times \overline{\mathcal{M}} \to \mathbb{R}^{d_1-1} \times \ldots \times \mathbb{R}^{d_L-1} \times \mathbb{R}$

$$F_2((W,b),(Y,c)) = \begin{bmatrix} (W_1^{\perp})^{\top} vec(Y_1) \\ \vdots \\ (W_L^{\perp})^{\top} vec(Y_L) \\ (b_1^{\perp})^{\top} c_1 \\ \vdots \\ (b_L^{\perp})^{\top} c_L \\ \sum_{i=1}^{L} log \frac{\langle vec(W_i), vec(Y_i)\rangle}{||vec(W_i)||_2^2} \\ log\left(\frac{\langle b_1, c_1\rangle}{||b_1||_2^2}\right) - log\left(\frac{\langle vec(W_1), vec(Y_1)\rangle}{||vec(W_1)||_2^2}\right) \\ log\left(\frac{\langle b_2, c_2\rangle}{||b_2||_2^2}\right) - log\left(\frac{\langle b_1, c_1\rangle}{||b_1||_2^2} \times \frac{\langle vec(W_1), vec(Y_1)\rangle}{||vec(W_1)||_2^2}\right) \\ \vdots \\ log\left(\frac{\langle b_L, c_L\rangle}{||b_L||_2^2}\right) - log\left(\frac{\langle b_{L-1}, c_{L-1}\rangle}{||b_{L-1}||_2^2} \times \frac{\langle vec(W_{L-1}), vec(Y_{L-1})\rangle}{||vec(W_{L-1})||_2^2}\right) \end{bmatrix}$$

For any $[X_1, \ldots, X_L, z_1, \ldots, z_L, x, u_1, \ldots u_L] \in \mathbb{R}^{d_1-1} \times \ldots \times \mathbb{R}^{d_L-1} \times \mathbb{R}^{n_1-1} \times \ldots \times \mathbb{R}^{n_L-1} \times \mathbb{R}^{L+1}$, we can define $(\tilde{Y}, \tilde{c})$ such that

$$vec(\tilde{Y}) = \left[W_1^{\perp} X_1 + \frac{x}{L} vec(Y_1), \ldots, W_L^{\perp} X_L + \frac{x}{L} vec(Y_L)\right]$$

$$\tilde{c} = \Bigg[b_1^{\perp} z_1 + \left(u_1 + \frac{x}{L}\right) c_1,$$

$$b_2^{\perp} z_2 + \left(u_2 + u_1 + \frac{2x}{L}\right) c_2,$$

$$\ldots,$$

$$b_L^{\perp} z_L + \left(u_L + u_{L-1} + \frac{Lx}{L}\right) c_L \Bigg]$$

which means, for points $(W, Y) \in graph(\sim)$:

$$DF_2((W,b),(Y,c))[0,(\tilde{Y},\tilde{c})] = [X_1, \ldots, X_L, z_1, \ldots, z_L,$$

$$x, u_1, \ldots, u_L]$$

This means that F_2 is a submersion at each point of graph$(\sim)$, and the set $F_2^{-1}(0) =$ graph$(\sim)$ is an embedded submanifold of $\overline{\mathcal{M}} \times \overline{\mathcal{M}}$. This concludes our proof that $\mathcal{M} = \overline{\mathcal{M}} / \sim$ is a quotient manifold.

3.7.2.1 Characterizing the Vertical Tangent Space of the Quotient Manifold

Let us now introduce a new invariant property of the equivalence class for network parameters with biases. First, for a point (W, b) in the space of parameters, we know that $W = (W_1, \ldots, W_L)$ and $b = (b_1, \ldots, b_L)$. For each layer, let us define $\tilde{W}_i \in \mathbb{R}^{d_i + n_i}$ as follows

$$\tilde{W}_i = \begin{cases} [\mathrm{vec}(W_i); \frac{b_i}{\|b_{i-1}\|}], & \text{if } i > 1, \\ [\mathrm{vec}(W_i); b_i], & \text{if } i = 1. \end{cases}$$

We then have that for each $(U,c) \in \approx^{-1}(\approx((W,b)))$ if $\prod_i \|\tilde{U}_i\|_2^2 = \prod_i \|\tilde{W}_i\|_2^2$, which is the invariant property of the equivalence class. We can also get a description of the tangent space of $\pi^{-1}(\pi((W,b)))$ from the following lemma.

Lemma 3.7.9 *The tangent space of $\approx^{-1}(\approx((W,b)))$ at (U,c) is $(\beta_1 U_1, ..., \beta_L U_L, \gamma_1 c_1, \dots, \gamma_L c_L)$ with $\sum_i \beta_i = 0$, $\beta_i = \gamma_i - \gamma_{i-1}$.*

Proof 3.7.10 *Consider the curves $(U_i(t), c_i(t)) \in \overline{\mathcal{M}}_i$ with $U_i(0) = U_i, c_i(0) = c_i$, we have*

$$\sum_i \log \|\tilde{U}_i(t)\|_2^2 = \sum_i \log \|\tilde{W}_i\|_2^2.$$

$$\implies \sum_i \log \left(\|U_i(t)\|_F^2 + \frac{\|c_i(t)\|^2}{\|c_{i-1}(t)\|^2} \right)$$

$$= \sum_i \log \left(\|W_i\|_F^2 + \frac{\|b_i\|^2}{\|b_{i-1}\|^2} \right).$$

Taking the derivative on both sides with respect to t gives

$$\sum_i \frac{1}{\|U_i(t)\|_F^2 + \frac{\|c_i(t)\|^2}{\|c_{i-1}(t)\|^2}} \times \Bigg(\langle \dot{U}_i(t), U_i(t) \rangle$$

$$+ \frac{\langle \dot{c}_i(t), c_i(t) \rangle}{\|c_{i-1}(t)\|^2} - \frac{\langle \dot{c}_{i-1}(t), c_{i-1}(t) \rangle \times \|c_i(t)\|^2}{\|c_{i-1}(t)\|^4} \Bigg) = 0.$$

It is clear that $\dot{U}_i(t), \dot{c}_i(t) = \beta_i U_i(t), \gamma_i c_i(t)$ with $\sum_i \beta_i = 0$ and $\beta_i = \gamma_i - \gamma_{i-1}$ satisfies the above equation. Therefore the tangent space $\mathcal{T}_{\approx^{-1}(\approx((W,b)))}(U,c)$ at U, c contains all tangent vectors $(\beta_1 U_1, ..., \beta_L U_L, \gamma_1 c_1, \dots, \gamma_L c_L)$ with $\sum_i \beta_i = 0$ and $\beta_i = \gamma_i - \gamma_{i-1}$.

References

LeCun, Yann, Yoshua Bengio, and Geoffrey Hinton (2015). “Deep learning”. In: *nature* 521.7553, p. 436.

Krizhevsky, Alex, Ilya Sutskever, and Geoffrey E Hinton (2012). “Imagenet classification with deep convolutional neural networks”. In: *Advances in neural information processing systems*, pp. 1097–1105.

Ren, Shaoqing, Kaiming He, Ross Girshick, and Jian Sun (2015). “Faster r-cnn: Towards real-time object detection with region proposal networks”. In: *Advances in neural information processing systems*, pp. 91–99.

Sutskever, Ilya, Oriol Vinyals, and Quoc V Le (2014). “Sequence to sequence learning with neural networks”. In: *Advances in neural information processing systems*, pp. 3104–3112.

Jean, Sébastien, Kyunghyun Cho, Roland Memisevic, and Yoshua Bengio (2014). “On using very large target vocabulary for neural machine translation”. In: *arXiv preprint arXiv:1412.2007*.

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”. In: *IEEE Signal processing magazine* 29.6, pp. 82–97.

Sainath, Tara N, Abdel-rahman Mohamed, Brian Kingsbury, and Bhuvana Ramabhadran (2013). “Deep convolutional neural networks for LVCSR”. In: *Acoustics, speech and signal processing (ICASSP), 2013 IEEE international conference on*. IEEE, pp. 8614–8618.

Chaudhari, Pratik, Anna Choromanska, Stefano Soatto, Yann LeCun, Carlo Baldassi, Christian Borgs, Jennifer Chayes, Levent Sagun, and Riccardo Zecchina (2016). “Entropy-sgd: Biasing gradient descent into wide valleys”. In: *arXiv preprint arXiv:1611.01838*.

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”. In: *arXiv preprint arXiv:1609.04836*.

Hochreiter, Sepp and Jürgen Schmidhuber (1995). “Simplifying neural nets by discovering flat minima”. In: *Advances in neural information processing systems*, pp. 529–536.

Hochreiter, Sepp and Jürgen Schmidhuber (1997). “Flat minima”. In: *Neural Computation* 9.1, pp. 1–42.

Dziugaite, Gintare Karolina and Daniel M Roy (2017). “Computing nonvacuous generalization bounds for deep (stochastic) neural networks with many more parameters than training data”. In: *arXiv preprint arXiv:1703.11008*.

Neyshabur, Behnam, Srinadh Bhojanapalli, David McAllester, and Nathan Srebro (2017). “A pac-bayesian approach to spectrally-normalized margin bounds for neural networks”. In: *arXiv preprint arXiv:1707.09564*.

Dinh, Laurent, Razvan Pascanu, Samy Bengio, and Yoshua Bengio (2017). “Sharp Minima Can Generalize for Deep Nets”. In: *Proceedings of the 34th International Conference on Machine Learning - Volume 70*. ICML’17. Sydney, NSW, Australia: JMLR.org, pp. 1019–1028. URL: http://dl.acm.org/citation.cfm?id=3305381.3305487.

Wang, Huan, Nitish Shirish Keskar, Caiming Xiong, and Richard Socher (2018). “Identifying Generalization Properties in Neural Networks”. In: *arXiv preprint arXiv:1809.07402*.

Novak, Roman, Yasaman Bahri, Daniel A Abolafia, Jeffrey Pennington, and Jascha Sohl-Dickstein (2018). “Sensitivity and generalization in neural networks: an empirical study”. In: *arXiv preprint arXiv:1802.08760*.

Cho, Minhyung and Jaehyung Lee (2017). “Riemannian approach to batch normalization”. In: *Advances in Neural Information Processing Systems*, pp. 5225–5235.

Hoffer, Elad, Ron Banner, Itay Golan, and Daniel Soudry (2018). “Norm matters: efficient and accurate normalization schemes in deep networks”. In: *arXiv preprint arXiv:1803.01814*.

Huang, Lei, Xianglong Liu, Bo Lang, and Bo Li (2017). “Projection based weight normalization for deep neural networks”. In: *arXiv preprint arXiv:1710.02338*.

Absil, P-A, Robert Mahony, and Rodolphe Sepulchre (2009). *Optimization algorithms on matrix manifolds*. Princeton University Press.

Absil, P-A, Robert Mahony, and Rodolphe Sepulchre (2004). “Riemannian geometry of Grassmann manifolds with a view on algorithmic computation”. In: *Acta Applicandae Mathematica* 80.2, pp. 199–220.

Li, Hao, Zheng Xu, Gavin Taylor, Christoph Studer, and Tom Goldstein (2018). "Visualizing the loss landscape of neural nets". In: *Advances in Neural Information Processing Systems*, pp. 6389–6399.

Salimans, Tim and Diederik P Kingma (2016). "Weight normalization: A simple reparameterization to accelerate training of deep neural networks". In: *Advances in Neural Information Processing Systems*, pp. 901–909.

Neyshabur, Behnam, Ruslan R Salakhutdinov, and Nati Srebro (2015). "Path-sgd: Path-normalized optimization in deep neural networks". In: *Advances in Neural Information Processing Systems*, pp. 2422–2430.

Chapter 4

Vision with Compressive Measurements

In this chapter we study the ability of deep network based computer vision systems (object classification and detection systems) to be able to work with compressive measurements of natural scenes and videos. The work in this chapter was earlier presented in the papers - Kwan et al., 2019c; Kwan et al., 2019b; Kwan et al., 2019a; Nair et al., 2018

4.1 Introduction

In the era of big data, Convolutional Neural Networks (CNNs) today have become one of the most powerful methods for visual tasks. They have achieved success in problems such as image classification (Krizhevsky, Sutskever, and Hinton, 2012; Simonyan and Zisserman, 2014; He et al., 2016) and object detection (Ren et al., 2015). Key to the success is the ability to learn rich feature hierarchies (Girshick et al., 2014) , with low-level features like edges and colors learned at lower layers, which are combined together in the higher layers to detect complex shapes and patterns in a fully-differentiable end-to-end framework.

Traditionally, CNNs are trained on fully observed images. However, in challenging real imaging scenarios sensing systems are often energy demanding or need to operate with limited bandwidth and exposure-time budgets like in (Zhang et al., 2016). Or exposed to high level noise in communication channels, the collected data suffers from severe missing information. A decision framework based on inference from partially observed data is needed for more energy-efficient hardware system design and robust performance for noisy environment.

Compressed sensing (CS) theory (Candes and Tao, 2006; Donoho, 2006) guarantees the exact recovery of signals at sub-Nyquist sampling rates with sparsity assumptions. It provides theoretical foundations for designing CS hardware systems and reconstructing signals from compressed measurements (Duarte et al., 2008). Consequently, efficient systems have been developed for generating compressed measurements for demanding applications include underwater sensing (Fazel, Fazel, and Stojanovic, 2011), drone-based imaging (Shetti and Vijayakumar, 2015; Zhang et al., 2016), satellite imaging (Michel et al., 2012), high-speed imaging (Sonoda et al., 2016; Reddy, Veeraraghavan, and Chellappa, 2011) and magnetic resonance imaging (Lustig, Donoho, and Pauly, 2007).

However, CS entails the need for slow iterative algorithms to perform recovery and/or inference on the sampled data. In addition, CS methods do not scale to the sizes of training data sets that the modern data deluge affords.

To compensate for these disadvantages, CNN-based approaches have been considered to deal with compressed measurements.

Reconstruction algorithms such as ReconNet (Kulkarni et al., 2016), DeepInverse (Mousavi and Baraniuk, 2017) and classification algorithm (Lohit, Kulkarni, and Turaga, 2016) have shown promising performances. However, in those cases the sampling/sensing operators are assumed to be known, fixed a-priori and tied to the particular neural network being trained. In addition, they hinge on the availability of very specialized hardware like a Digital micro-mirror (DMD) array in order to allow efficient sensing implementations.

In this chapter, we attempt to overcome these difficulties by directly performing classification on partially observed measurements. The test images here are various fraction of the image scene's pixels chosen randomly, which model measurements from CS hardware and partial observations due to noise. We demonstrate the sensitivity of pre-trained convolutional neural networks, which fail miserably with only a small portion of missing pixels and propose a framework to overcome it through making the network learn from fully-observed and compressed images in the training procedure. We also empirically verify that our approach generalizes to unseen observation ratios without retraining the network.

Our framework is low-cost, efficient, and hardware friendly. It has several advantages: *(i)* Reconstruction-free in discriminative applications; *(ii)* Robust to changes in the partial observation mask; *(iii)* Retraining-free and generalizable to test data with unseen partial observation ratios; *(iv)* Transfers across visual tasks. *(v)* Efficiently deals with missing and incomplete data as long as

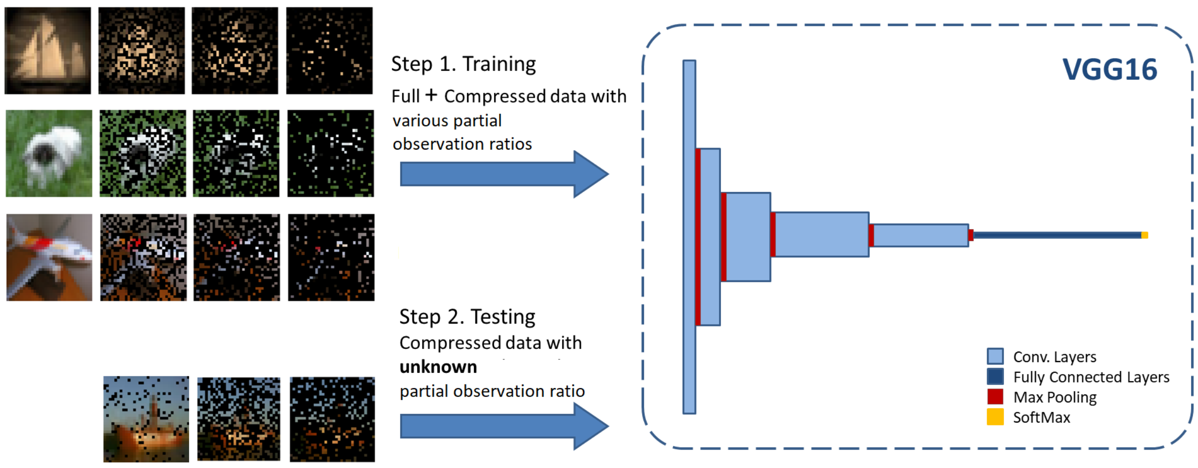

Step 1. Training
Full + Compressed data with various partial observation ratios
Step 2. Testing
Compressed data with **unknown** partial observation ratio
VGG16
Conv. Layers
Fully Connected Layers
Max Pooling
SoftMax

Figure 4.1: Overview of our framework in image classification. During training, we input full images and images with missing pixel ratios of .5, .25, .125 to a VGG16 (Simonyan and Zisserman, 2014) network. The test data to the network with partial observation ratio randomly generated between $(0, 1]$

the label information is correct.

4.2 Related work

4.2.1 Reconstructing CS measurements via CNNs

The CS measurements $\mathbf{y} \in \mathbb{R}^m$ of a signal $\mathbf{x} \in \mathbb{R}^n$, are generated using $\mathbf{y} = \mathbf{\Phi x}$ with a smaller dimension than the signal dimension, where the sensing matrix $\mathbf{\Phi} \in \mathbb{R}^{m \times n}$ is a random matrix (Candes and Tao, 2006). Recent work such as ReconNet (Kulkarni et al., 2016) and DeepInverse (Mousavi and Baraniuk, 2017) propose using CNNs to perform reconstruction from CS measurements. In ReconNet (Kulkarni et al., 2016), CNNs are then employed to reconstruct the CS measurements of each image block. All reconstructed blocks are then arranged and fed into a denoiser. DeepInverse (Mousavi and Baraniuk, 2017) on the other hand proposes using CNNs to learn the inverse transformation of to invert CS measurements $\mathbf{y}$ to signals $\mathbf{x}$.

There are several drawbacks to either employing a reconstruction algorithm before CNNs for partially-observed data or incorporating the reconstruction network into the entire framework. First, reconstruction is power-consuming. Second, the reconstruction network does not generalize well for test images with unseen and various partial observation ratios.

4.2.2 Classification on CS measurements using CNNs

To the best of our knowledge, the only work that proposes a classification algorithm on CS measurements using CNNs is (Lohit, Kulkarni, and Turaga, 2016). Instead of reconstructing images from compressive measurements $\boldsymbol{y}$ before feeding to CNNs, they perform a projection on the measurements $\mathbf{\Phi}^T\boldsymbol{y}$, and which is then resized into the original image size. Their framework performs well on MNIST and ImageNet with low measurement rates.

Their work demonstrates the promise of classification directly on CS Measurements using CNNs. However, the huge disadvantage of this framework is that the sensing matrix is fixed. In several image sensing models, the sensing operation of training data varies each time. Also, it does not generalize to new unseen sampled data without re-training the network.

4.3 Method: Extracting information from partially observed images

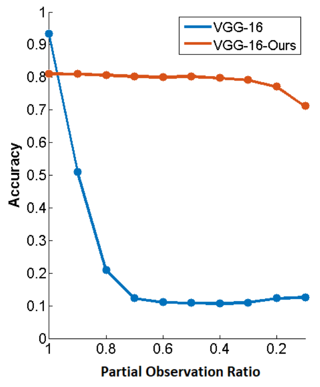
We propose a framework to extract information from visual data with an unknown fraction of pixels missing using CNNs, without performing reconstruction or re-training the neural network for every possible partial observation ratio.

We first generate partially observed training data corresponding to k ratios between 0 and 1. In this chapter, we use the original fully-observed data, along with data observed at three ratios of $0.5, 0.25$ and 0.125. We then train neural network with the enlarged training set. The ratio of the randomly observed pixels in the testing data need not match the ratios used during training, so as the random observation masks. An overview of our framework in image classification is shown in Figure 4.1.

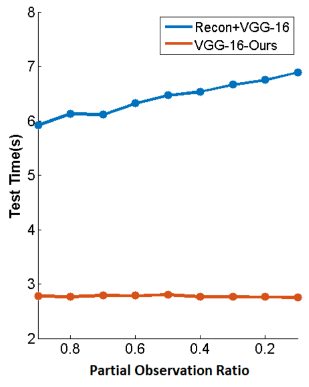
In the image classification task, the neural network that we use is the VGG-16 (Simonyan and Zisserman, 2014) network. Our proposed framework is also tested on object detection, in which Faster-RCNN, based on VGG-16 features, is employed.

4.4 Experiments

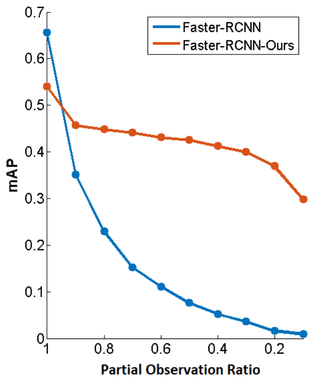
In this section, we test with our method on two common visual tasks - image classification and object detection. For image classification, we evaluate our network on the standard CIFAR-10 (Krizhevsky and Hinton, 2009) dataset, while we use the Pascal-VOC 2007 ("The PASCAL Visual Object Classes



(a) Averaged classification accuracies for VGG-16 (Simonyan and Zisserman, 2014) network and our method denoted as VGG-16-Ours



(b) Testing times on data for reconstruction plus VGG-16 network (Recon+VGG-16) and our reconstruction-free method



(c) Mean Average Precision(mAP) for Faster-RCNN and our framework: Faster-RCNN-Ours

Figure 4.2: (a) and (c) Classification accuracy and object detection performance for classical CNNs and ours with various partial observation ratios. (b) Testing times for doing reconstruction algorithm vs ours with various observation ratios

Challenge 2007 (VOC2007) Results") dataset for testing object detection.

4.4.1 Image Classification

The CIFAR-10 (Krizhevsky and Hinton, 2009) dataset contains 60000 images equally split between 10 object categories, with 50000 images marked as training and 10000 as test. Each image has 32×32 pixels.

We first train the VGG-16 CNN with default parameters from (Simonyan and Zisserman, 2014) on the dataset with full images, and as we see from Fig. 4.2a and Table 4.1, the classification accuracy is 0.93. We call this model 'VGG-16' henceforth. However, testing it on partially observed data, the classification accuracy drops sharply. To remedy this, we retrain the VGG-16 CNN network on full images as well as partially observed images. We used SGD

Partial Observation Ratios	**1.0**	**0.9**	**0.8**	**0.7**	**0.6**	**0.5**
VGG-16	**0.93**	0.51	0.21	0.12	0.11	0.11
VGG-16-Ours	0.81	0.81	0.81	0.80	0.80	0.80
Recon+VGG-16	**0.93**	**0.93**	**0.93**	**0.92**	**0.91**	**0.89**
Recon+VGG-16-Ours	0.81	0.81	0.81	0.81	0.81	0.80

Partial Observation Ratios	**0.4**	**0.3**	**0.2**	**0.1**	**Random**
VGG-16	0.11	0.11	0.12	0.13	0.19
VGG-16-Ours	0.80	**0.79**	**0.77**	**0.71**	**0.76**
Recon+VGG-16	**0.86**	**0.79**	0.65	0.35	-
Recon+VGG-16-Ours	0.80	0.78	0.74	0.62	-

Table 4.1: Averaged classification accuracies with various partial observationratios for four different methods: *(i)* VGG-16; *(ii)* VGG-16-Ours; *(iii)* Recon+VGG-16: reconstruct first and then use VGG-16; and *(iv)* Recon+VGG-16-Ours: reconstruct first and then use the network trained by our method. “Random” denotes the case that each test datum is randomly partially-observed by a ratio generated from $(0, 1]$ uniformly at random. The two dashes $(-)$ in the last column denote that the experiments are not performed since the reconstruction method is not robust to unknown partial observation ratios.

with momentum$= 0.9$, learning rate$= 0.1$, learning rate decay$= 10^{-6}$, batch size$= 128$ for 250 epochs with data augmentation through random translations, flips and rotations. We noted that only three partial observation ratios of $0.5, 0.25$ and 0.125 were sufficient to ensure robustness to such corruption as displayed in Fig. 4.2a. These three partial observation ratios were chosen empirically as a trade-off between magnitude of training data required and robustness to random missing. This is interesting, as it suggests the CNN has learned to generalize to randomly missing data. We term the network trained as such ‘VGG-16-Ours’ for purposes of discussion.

We see from Fig. 4.2a and Table 4.1 that as as we miss more data, performance degrades. However, even in the challenging scenario of having available a mere 10% of pixels, the network is achieves a classification accuracy of 0.71.

In order to compare our solution with the standard paradigm of reconstruct-then-classify, we train a set of 9 de-noising convolutional autoencoders on the CIFAR-10 dataset, one for each partial observation ratio from $0.1, 0.2, ...0.9$. We then passed test images with each of those ratios being observed through the corresponding autoencoder to reconstruct it, and fed the output to a pre-trained VGG-16 network. We term this experimental pipeline 'Recon+VGG-16' and refer to it as the same. As seen from the results in Table 4.1, this pipeline outperforms VGG-16-Ours, trained to classify on the partially-observed data directly at high observation ratios. However, for the more challenging cases of $0.3., 0.2$ and especially for the 0.1 case, VGG-16-Ours performs just as well if not substantially better than the standard reconstruction pipeline. Keeping our end-goal of sensing as little as we can get away with in mind, our results suggest that discrimination should be performed directly on the compressed data.

In addition, we also study the time required for the Recon+VGG-16 processing pipeline versus VGG-16-Ours in Fig. 4.2b. As both the networks are feed-forward neural networks, they each process the data by repeatedly applying basic arithmetic operations like multiplication, addition and a simple non-linearity on the data, once trained. This means testing performance for both pipelines is quite fast and the room of improvement is small. However, we note that direct classification on the compressed data is accomplished twice as fast (taking only 2.78 seconds averaging across all partial observation ratios) for classifying all 10000 test examples in CIFAR-10 as compared to Recon+VGG-16 (which takes 6.42 seconds averaging across all partial observation ratios). This obvious advantage stems from skipping the unnecessary step of reconstruction, speeding up the imaging as well as classification processes.

In order to better understand the behavior of VGG-16-Ours, we also used it to classify compressed data after reconstruction. Interestingly, as the results in Table 4.1 confirm, it would appear that the act of training VGG-16-Ours on partially observed data as well (which we term 'Recon+VGG-16-Ours' in the table) has made it more robust to perturbations in the input space, leading to a much higher classification performance in the challenging 0.1 observation ratio case (obtaining 0.62 classification accuracy) versus just passing the partially-observed data through the convolutional autoencoder and classifying it using a VGG-16 network trained only on fully-sampled data (yielding just 0.35 classification accuracy). The reconstructed data from the autoencoder loses a lot of high frequency information. This experiment suggests that including data with various observatio ratios has the added bonus of making a neural network robust to blur.

For the last column in Table 4.1, we randomly take a fraction of pixels in each test image by an unknown fraction $s_i \in (0, 1]$, and then passed the test images through VGG-16 and VGG-16-Ours. As expected, VGG-16-Ours obtained a much higher classification accuracy (a respectable 0.76, close to the average of the accuracies on the different partial observation ratios previously tested) on the test data than VGG-16 (0.19). We did not run Recon+VGG-16 and Recon+VGG-16-Ours experiments on this test set as the partial observation ratio was not constrained to match with the 9 partial observation ratios for which we had trained the autoencoders.

4.4.2 Object Detection

The Pascal VOC 2007 detection dataset ("The PASCAL Visual Object Classes Challenge 2007 (VOC2007) Results") contains images corresponding to 20 different object categories as part of various natural scenes, with close to 5000 images provided for training and cross-validation, with approximately another 5000 provided for testing.

Aiming to understand if the phenomenon of CNNs learning to handle arbitrary partial observation ratios extends to tasks besides object classification, we train a Faster RCNN network on the Pascal VOC 2007 dataset for object detection. We use the mean average precision (mAP) as our evalutaion metric. Details for the measure are contained in the original paper (Ren et al., 2015). Similar to Section 4.4.1, we initially train the network purely on fully-observed images. We term this trained network 'Faster-RCNN'. Then, we train another model with the same Faster-RCNN architecture on the fully-observed images as well as partially-observed images at observation ratios of 0.5, 0.25 and 0.125. We then tested both networks on object detection from partially-observed test images of various unseen partial observation ratios.

The results of the experiments are displayed in Fig. 4.2c. We note the trends here mirror those of the classification case, with Faster-RCNN's performance dropping quickly in the presence of random missing , while the performance of Faster-RCNN-Ours remains much more stable. In addition, we again observe that Faster-RCNN-Ours generalizes to unseen partial observation ratios here as well.

4.5 Target Tracking and Classification using a Compressive Sensing Camera

One of the motivations behind our study of deep networks to be able to handle compressive measurements in the previous sections, is our desire to use Pixelwise Coded Exposure (PCE) cameras to track targets in the videos that they collect. Figure 4.3 illustrates the differences between a conventional video sensing scheme and PCE, where random spatial pixel activation is combined with fixed temporal exposure duration. First, conventional cameras capture frames at certain frame rates, such as 30 frames per second. In contrast, the PCE camera captures a compressed frame called motion coded image over a fixed period of time (T_v). For example, a user can compress 30 conventional frames into a single motion coded frame. This will yield significant data compression ratio. Second, the PCE camera allows a user to use different exposure times for different pixel locations. For low lighting regions, more exposure times can be used and for strong light areas, short exposure can be exerted. This will allow high dynamic range. Moreover, power can also be saved via low sampling rate in the data acquisition process. As shown in Fig. 4.3, one conventional approach to using the motion coded images is to apply sparse reconstruction to reconstruct the original frames and this process may be very time-consuming.

Suppose the video scene is contained in a data cube $\mathbf{X} \in \mathbb{R}^{M \times N \times T}$ where

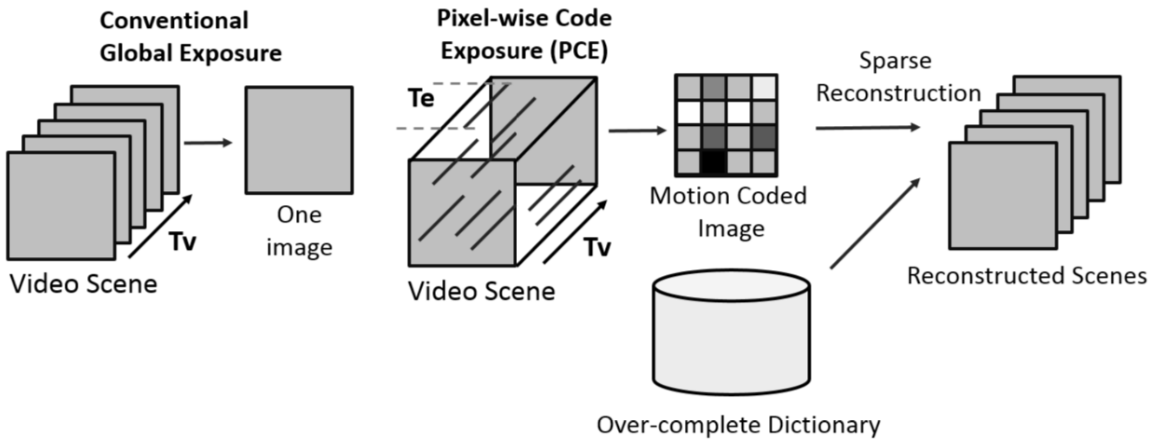


Figure 4.3: Conventional camera vs. pixel-wise coded exposure (PCE) compressed image/video sensor

$M \times N$ is the image size and T is the number of frames. A sensing data cube is defined by $\mathbf{S} \in \mathbb{R}^{M \times N \times T}$ which contains the exposure times for pixel located at (m, n, t). The value of $\mathbf{S}(m, n, t)$ is 1 for frames $t \in [t_{\text{start}}, t_{\text{end}}]$ and 0 otherwise. $[t_{\text{start}}, t_{\text{end}}]$ denotes the start and end frame numbers for a particular pixel. The measured coded aperture image $\mathbf{Y} \in \mathbb{R}^{M \times N}$ is obtained by:

$$\mathbf{Y}(m, n) = \sum_{t=1}^{T} \mathbf{S}(m, n, t) \cdot \mathbf{X}(m, n, t)$$

Instead of doing sparse reconstruction on PCE images or frames, our scheme directly acts on the PCE or coded aperture images, which contain raw sensing measurements without the need for any reconstruction effort. Utilizing raw measurements has several challenges. First, moving targets may be smeared if the exposure times are long. Second, there are also missing pixels in the raw measurements because not all pixels are activated during the data collection process. Third, there are much fewer frames in the raw video because many original frames are compressed into a single coded frame.

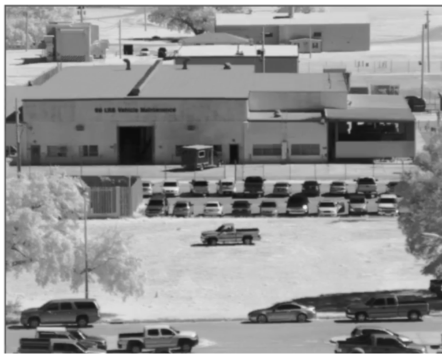
In this section we will show that by simulating the measurements that should be produced by the PCE-based compressive sensing (CS) sensor, we can show that detecting, tracking, and even classifying moving objects of interest in the scene is entirely feasible with a minor sacrifice in discrimination accuracy.

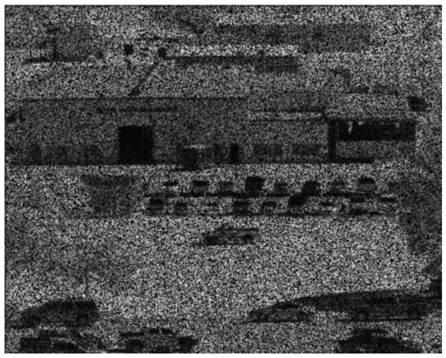
4.5.1 Task and Dataset

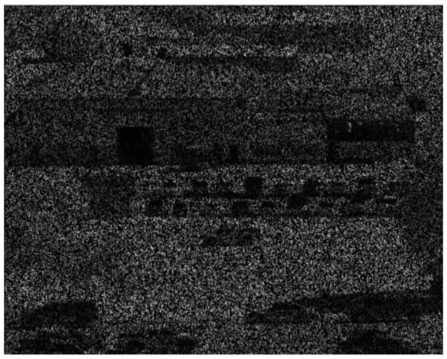
We have a custom dataset of videos of three kinds of trucks (Ram, Silverado, and Frontier) driving in a parking lot. All of the videos are short wave IR videos, each about 1 min long, with 30 frames/sec. This means we have 1800 frames per video. There are two videos for each truck, each recorded on a different day. We used all the frames from one of the days for training and the other day's videos for testing. Our task is to detect and track the moving truck through a video, as well as identify the type of truck in the video. A sample frame from this video and its subsampled versions are shown in Fig 4.4

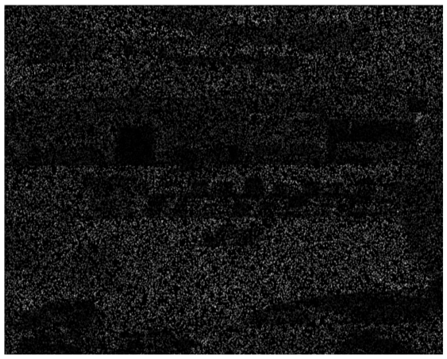
In addition to the above dataset, we have two other videos where all three trucks travel back and forth between a parking lot and a remote location. These two videos are more challenging because they have multiple targets, and the targets change in size as they travel. In order to adapt our systems for this task, we fine-tuned a network trained on the previous dataset to one of the videos here, and used the other video to obtain test results.

Figure 4.4: Sample Frame of a Frontier truck from the training dataset


(a) Sample Frame


(b) Sample Frame with 50% PCE measurements


(c) Sample Frame with 25% PCE measurements


(d) Sample Frame with 12.5% PCE measurements

4.5.2 Deep Networks used for Tracking and Classification

We perform tracking by detection, meaning we perform object detection on each motion coded image to achieve tracking across a video. Object detection on each frame is performed using a deep network, specifically the You Only Look Once (YOLO) architecture (Redmon and Farhadi, 2016). The YOLO tracker is fast and has similar performance to the Faster R-CNN (Ren et al., 2015). YOLO has 24 convolutional layers and 2 fully connected layers. The inputs are resized to 448×448, and the output is $7 \times 7 \times 30$. We have used YOLOv2 because it is more accurate than YOLO version 1.

Since YOLO's built-in classifier did not perform too well on our task, we decided to use another network to perform classification. The ResNet-18 model is an 18-layer convolutional neural network (CNN) that has the advantage of avoiding performance saturation and/or degradation when training deeper layers, which is a common problem among other CNN architectures. The ResNet-18 model avoids the performance saturation by implementing an identity short cut connection, which skips one or more layers and learns the residual mapping of the layer rather than the original mapping.

Our pipeline was implemented as follows - YOLO was used to determine where, in each frame, the trucks were located. YOLO generated bounding boxes for those trucks and that data were used to crop the trucks from the image. The cropped trucks would be fed into the ResNet-18 for classification, and classification results were generated.

4.5.3 Tracking and classification results

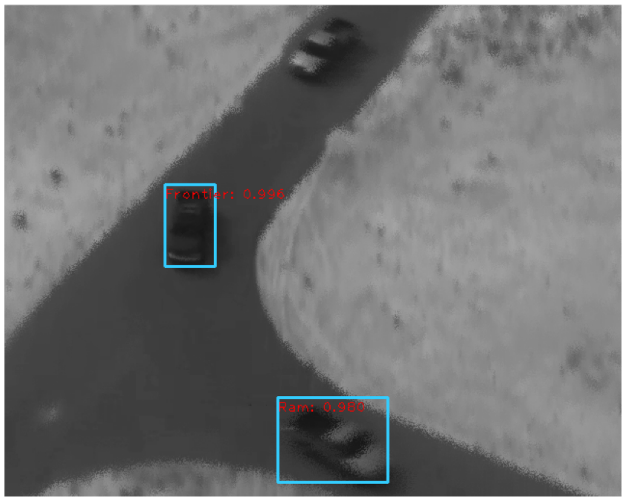
We trained our deep networks using SGD (learning rate = 0.001) with momentum (0.9) for 2000 epochs on the dataset of individual truck videos with complete frames and 12.5% subsampled PCE frames. We started with a models pretrained on the Pascal VOC dataset for the YOLO network and Imagenet for the ResNet-18 network. We then tested our model on the held out videos, with complete as well as PCE frames with subsampling rates of 6.25%, 12.5%, 25%, and 50%. To evaluate detections we measured the following:

- Detection Rate - Fraction of the frames of the videos in which a detection was made
- DICE score - given a detection A and a ground truth A^*, the DICE score is defined as $\frac{2|A \cap A^*|}{|A|+|A^*|}$
- Centroid distance - ℓ_2 distance between the centroids of the detection and the ground truth, in pixels.
- Classification accuracy

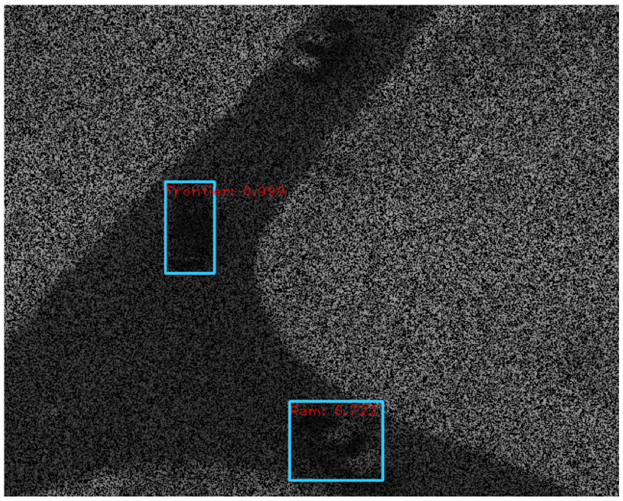
These results are presented in Tables 4.2, 4.3, 4.4 and sample detection frames are shown in Fig. 4.5.

Figure 4.5: Detection and classification on a sample frame

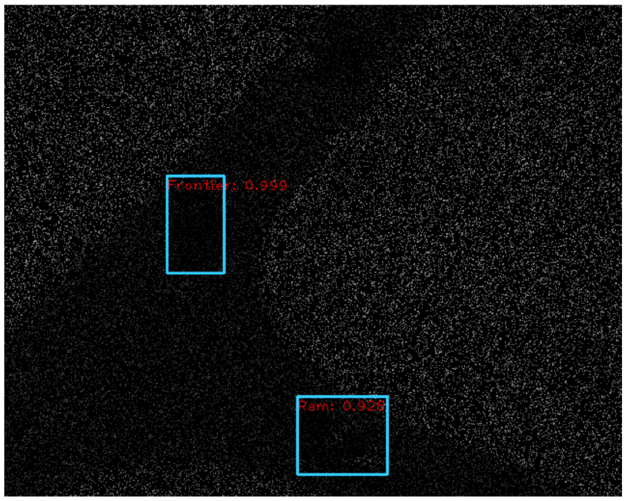
(a) 100% PCE measurements



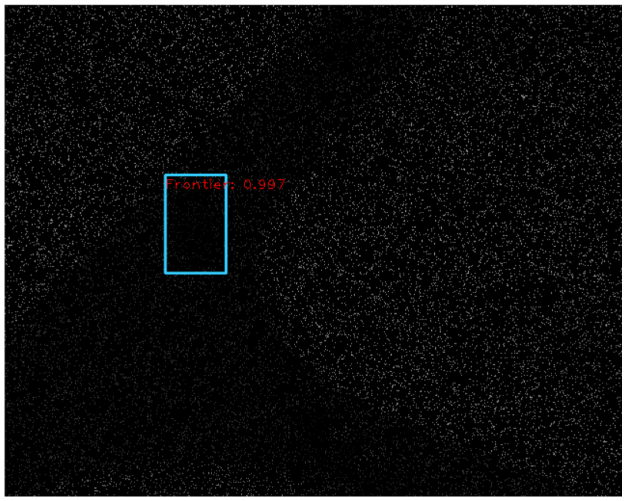
(b) 50% PCE measurements



(c) 25% PCE measurements



(d) 12.5% PCE measurements



Coded Aperture Fraction	**Silverado**	**Ram**	**Frontier**
6.25%	63%	65%	68%
12.5%	96%	95%	97%
25%	93%	93%	95%
50%	94%	93%	95%
100%	97%	98%	98%

Table 4.2: Detection Rates - Fraction of frames in which a target was detected

Coded Aperture Fraction	**DICE score**	**Centroid ℓ_2 distance** (in pixels)
6.25%	0.782	8.22
12.5%	0.834	7.75
25%	0.816	8.04
50%	0.801	8.13
100%	0.852	7.52

Table 4.3: Detection Scores, averaged over all frames in which a target was detected

4.6 Conclusion

This chapter presents an efficient, reconstruction-free training paradigm to extract information from sparsely-sensed images, overcoming the sensitivity that CNNs naturally have to such input mismatches. Moreover, the proposed method generalizes to different, unseen, arbitrary partial observation ratios without retraining. Our method outperforms the pre-trained CNNs and reconstruction-first-classify-later technique in challenging cases with small observation ratios.

Investigation of the role of missing information may play in making a network more robust to adversarial interference is an interesting open question. Finally, extending our framework to handle missing/incomplete or partially corrupted data and sensor failure is a possible future research direction.

Coded Aperture Fraction	**Silverado**	**Ram**	**Frontier**
6.25%	65%	72%	74%
12.5%	79%	94%	96%
25%	81%	95%	94%
50%	74%	93%	83%
100%	95%	98%	100%

Table 4.4: Classification Rates - Fraction of frames in which the target was correctly classified

References

Kwan, Chiman, Bryan Chou, Jonathan Yang, Akshay Rangamani, Trac Tran, Jack Zhang, and Ralph Etienne-Cummings (2019c). “Target tracking and classification using compressive sensing camera for SWIR videos”. In: *Signal, Image and Video Processing*, pp. 1–9.

Kwan, Chiman, Bryan Chou, Jonathan Yang, Akshay Rangamani, Trac Tran, Jack Zhang, and Ralph Etienne-Cummings (2019b). “Target tracking and classification using compressive measurements of MWIR and LWIR coded aperture cameras”. In: *Journal of Signal and Information Processing* 10.3, pp. 73–95.

Kwan, Chiman, Bryan Chou, Jonathan Yang, Akshay Rangamani, Trac Tran, Jack Zhang, and Ralph Etienne-Cummings (2019a). “Deep Learning-Based Target Tracking and Classification for Low Quality Videos Using Coded Aperture Cameras”. In: *Sensors* 19.17, p. 3702.

Nair, Arun Asokan, Luoluo Liu, Akshay Rangamani, Peter Chin, Muyinatu A. Lediju Bell, and Trac D. Tran (2018). “Reconstruction-free deep convolutional neural networks for partially observed images”. In: *2018 IEEE Global Conference on Signal and Information Processing (GlobalSIP)*. IEEE.

Krizhevsky, Alex, Ilya Sutskever, and Geoffrey E Hinton (2012). “Imagenet classification with deep convolutional neural networks”. In: *Advances in Neural Information Processing Systems*, pp. 1097–1105.

Simonyan, Karen and Andrew Zisserman (2014). “Very deep convolutional networks for large-scale image recognition”. In: *arXiv preprint arXiv:1409.1556*.

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*, pp. 770–778.

Ren, Shaoqing, Kaiming He, Ross Girshick, and Jian Sun (2015). “Faster R-CNN: Towards real-time object detection with region proposal networks”. In: *Advances in Neural Information Processing Systems*, pp. 91–99.

Girshick, Ross, Jeff Donahue, Trevor Darrell, and Jitendra Malik (2014). "Rich feature hierarchies for accurate object detection and semantic segmentation". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 580–587.

Zhang, Jie, Tao Xiong, Trac Tran, Sang Chin, and Ralph Etienne-Cummings (2016). "Compact all-CMOS spatiotemporal compressive sensing video camera with pixel-wise coded exposure". In: *Optics Express* 24.8, pp. 9013–9024.

Candes, Emmanuel J and Terence Tao (2006). "Near-optimal signal recovery from random projections: Universal encoding strategies?" In: *IEEE Transactions on Information Theory* 52.12, pp. 5406–5425.

Donoho, David L (2006). "Compressed sensing". In: *IEEE Transactions on Information Theory* 52.4, pp. 1289–1306.

Duarte, Marco F, Mark A Davenport, Dharmpal Takhar, Jason N Laska, Ting Sun, Kevin F Kelly, and Richard G Baraniuk (2008). "Single-pixel imaging via compressive sampling". In: *IEEE Signal Processing Magazine* 25.2, pp. 83–91.

Fazel, Fatemeh, Maryam Fazel, and Milica Stojanovic (2011). "Random access compressed sensing for energy-efficient underwater sensor networks". In: *IEEE Journal on Selected Areas in Communications* 29.8, pp. 1660–1670.

Shetti, Karan and Asha Vijayakumar (2015). "Evaluation of compressive sensing encoding on AR drone". In: *Signal and Information Processing Association Annual Summit and Conference (APSIPA), 2015 Asia-Pacific*. IEEE, pp. 204–207.

Michel, Julien, Gwendoline Blanchet, Julien Malik, and Rosario Ruiloba (2012). "Compressed sensing for earth observation with high resolution satellite imagery". In: *Geoscience and Remote Sensing Symposium (IGARSS), 2012 IEEE International*. IEEE, pp. 412–415.

Sonoda, Toshiki, Hajime Nagahara, Kenta Endo, Yukinobu Sugiyama, and Rinichiro Taniguchi (2016). "High-speed imaging using CMOS image sensor with quasi pixel-wise exposure". In: *Computational Photography (ICCP), 2016 IEEE International Conference on*. IEEE, pp. 1–11.

Reddy, Dikpal, Ashok Veeraraghavan, and Rama Chellappa (2011). "P2C2: Programmable pixel compressive camera for high speed imaging". In: *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*. IEEE, pp. 329–336.

Lustig, Michael, David Donoho, and John M Pauly (2007). “Sparse MRI: The application of compressed sensing for rapid MR imaging”. In: *Magnetic Resonance in Medicine* 58.6, pp. 1182–1195.

Kulkarni, Kuldeep, Suhas Lohit, Pavan Turaga, Ronan Kerviche, and Amit Ashok (2016). “Reconnet: Non-iterative reconstruction of images from compressively sensed measurements”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 449–458.

Mousavi, Ali and Richard G Baraniuk (2017). “Learning to invert: Signal recovery via deep convolutional networks”. In: *Acoustics, Speech and Signal Processing (ICASSP), 2017 IEEE International Conference on*. IEEE, pp. 2272–2276.

Lohit, Suhas, Kuldeep Kulkarni, and Pavan Turaga (2016). “Direct inference on compressive measurements using convolutional neural networks”. In: *Image Processing (ICIP), 2016 IEEE International Conference on*. IEEE, pp. 1913–1917.

Krizhevsky, Alex and Geoffrey Hinton (2009). “Learning multiple layers of features from tiny images”. In:

Everingham, M., L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. “The PASCAL Visual Object Classes Challenge 2007 (VOC2007) Results”. In:

Redmon, Joseph and Ali Farhadi (2016). “YOLO9000: Better, Faster, Stronger”. In: *arXiv preprint arXiv:1612.08242*.

Chapter 5

Summary and Future Work

Machine learning techniques in general and deep learning in particular are gaining popularity as a approaches to solve to all sorts of problems. Deep learning is being applied with great success to a number of different problems, but the success depends on careful engineering of features, choice of architecture, choice of training schemes, etc. It is critical to understand why and how deep networks work, and when they are the best approach for a problem.

In this dissertation we saw results that helped us understand the representations that neural networks learn and which solutions of deep network training are more likely to generalize. We also saw some applications of deep learning to vision problems with subsampled measurements.

In Chapter 2 we analyzed the loss landscape of autoencoders and were able to establish connections between autoencoders and *Sparse Coding* or *Dictionary Learning*. We showed that under a sparse coding generative model, the landscape of the squared reconstruction error of a ReLU autoencoder has a critical point at the ground truth dictionary. Simulations also tell us that if we start a gradient descent algorithm far away from the ground truth dictionary, we end up close to it after enough iterations.

In Chapter 3 we studied the "flat minima" problem. While the idea that flatter minima could generalize better has been around for a long time (Hochreiter and Schmidhuber, 1997), it was recently shown (Dinh et al., 2017) that for deep networks with positively homogenous activation functions (like ReLU) quantitative measures of "flatness" could be made arbitrarily large or small through a simple rescaling of the deep network. Using techniques from manifold geometry, we proposed a flatness metric that is invariant to these rescalings. We then applied this technique to compare minima obtained using large-batch and small-batch gradient based methods, and were able to empirically confirm the observation that "flatter minima" generalize better. Our work is one of the first to consider the space of deep network parameters as a differentiable quotient manifold rather than a Euclidean space.

Finally, in Chapter 4 we presented a deep learning pipeline to train networks to solve vision problems from compressed measurements. We built an object detection and tracking system based on deep networks that can work with a custom image sensorthat collects compressive measurements of scenes. We studied some training schemes that allow us to adapt deep networks for object detection from natural images to our setting (Nair et al., 2018). We also presented results on tracking objects in video sequences in specialized settings, using compressive measurements that simulate the sensor presented in (Zhang et al., 2016)

5.1 Future Work

- **Understanding Recurrent Neural Networks (RNNs):** Recurrent Neural Networks have been shown to be successful in many tasks like speech recognition and machine translation. While they have also been applied to time series data with varying degrees of success, we still do not have insight into the kinds of problems that can be solved using recurrent networks. There is some evidence that we can learn linear dynamical systems using linear RNNs Hardt, Ma, and Recht, 2018, while the behavior of more complicated, nonlinear models is not yet understood. We can use an approach similar to the one used to study autoencoders in order to understand RNNs.

- **Sparse Neural Networks:** An intriguing property of neural networks is that they can be compressed to a fraction of their original size, while retaining performance levels similar to that of the original network Han, Mao, and Dally, 2015. There is also some evidence that the compressibility of a deep network is related to its generalization Zhou et al., 2018. One promising direction in understanding deep networks is exploring whether the connections between compressibility and generalization also extend to sparsity and generalization. This would have implications for practical applications of deep learning in resource constrained settings, since sparse neural networks are smaller, and computationally cheaper to implement.

  Current approaches to the compression of neural networks involve training a large deep network model first, followed by retraining a sparser model after removing connections that are not salient. One can also investigate approaches to training sparse neural networks “from scratch", that is, without having to train the large models first. Training techniques that are not based on gradient based algorithms, like the Alternating Direction Method of Multipliers (ADMM) (Taylor et al., 2016), are one possible line of research.

- **Training Deep Networks on the Manifold:** The quotient manifold of parameters that we presented in Chapter 3 is a promising view for not just obtaining invariant measurements of the flatness of deep network minima, but also training deep networks. Instead of performing stochastic gradient descent type algorithms in the Euclidean space of parameters, we would like to learn deep networks by optimizing the objective function on the quotient manifold of neural network parameters. It would also be interesting to explore the connections between training deep networks on manifolds and techniques like Batch Normalization and Weight Normalization.

- **Neural Networks for Coded Aperture Video Reconstruction:** While we can use traditional dictionary learning and sparse recovery based methods to reconstruct video sequences from Pixelwise Coded Exposure (PCE) measurements, these dictionaries and sparse recovery algorithms are limited to the particular frame rate, exposure time, and other parameters of the measurement process. In order to reconstruct these video sequences in a number of different settings, we can explore deep learning based solutions. Recurrent architectures might be well suited to this approach of reconstructing videos agnostic to the frame rate.

References

Hochreiter, Sepp and Jürgen Schmidhuber (1997). "Flat minima". In: *Neural Computation* 9.1, pp. 1–42.

Dinh, Laurent, Razvan Pascanu, Samy Bengio, and Yoshua Bengio (2017). "Sharp minima can generalize for deep nets". In: *arXiv preprint arXiv:1703.04933*.

Nair, Arun Asokan, Luoluo Liu, Akshay Rangamani, Peter Chin, Muyinatu A. Lediju Bell, and Trac D. Tran (2018). "Reconstruction-free deep convolutional neural networks for partially observed images". In: *2018 IEEE Global Conference on Signal and Information Processing (GlobalSIP)*. IEEE.

Zhang, Jie, Tao Xiong, Trac Tran, Sang Chin, and Ralph Etienne-Cummings (2016). "Compact all-CMOS spatiotemporal compressive sensing video camera with pixel-wise coded exposure". In: *Optics express* 24.8, pp. 9013–9024.

Hardt, Moritz, Tengyu Ma, and Benjamin Recht (2018). "Gradient descent learns linear dynamical systems". In: *The Journal of Machine Learning Research* 19.1, pp. 1025–1068.

Han, Song, Huizi Mao, and William J Dally (2015). "Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding". In: *arXiv preprint arXiv:1510.00149*.

Zhou, Wenda, Victor Veitch, Morgane Austern, Ryan P Adams, and Peter Orbanz (2018). "Compressibility and Generalization in Large-Scale Deep Learning". In: *arXiv preprint arXiv:1804.05862*.

Taylor, Gavin, Ryan Burmeister, Zheng Xu, Bharat Singh, Ankit Patel, and Tom Goldstein (2016). "Training neural networks without gradients: A scalable admm approach". In: *International Conference on Machine Learning*, pp. 2722–2731.

Vita

Akshay Rangamani was born in St.Louis, MO, USA in 1992, but spent the bulk of his childhood and adolescent years in Chennai, Tamil Nadu, India. He received the B.Tech degree in Electrical Engineering from the Indian Institute of Technology, Madras in 2013 and the M.S.E degree in electrical and computer engineering from the Johns Hopkins University, Baltimore, MD, USA, in 2015. Currently, he is pursuing the Ph.D. degree in Electrical and Computer Engineering at Johns Hopkins University. Since 2014, he has been working in the Digital Signal Processing Lab at Johns Hopkins University where his research is focusing on understanding deep learning techniques from a theoretical perspective, as well as applying them to signal processing problems. Over the course of his Ph.D. degree, he has worked as an intern at Draper Laboratories, Cambridge, MA, Uplevel Security, New York, NY, and IBM T.J. Watson Research Center, Yorktown Heights, NY.