2020-07-30

Optimal Learning Rates for Neural Networks

Tyler Moncur
Brigham Young University

Optimal Learning Rate for Neural Networks

Tyler Moncur

A thesis submitted to the faculty of
Brigham Young University
in partial fulfillment of the requirements for the degree of

Master of Science

Tyler Jarvis, Chair
Mark Hughes
David Wingate

Department of Mathematics

Brigham Young University

ABSTRACT

Optimal Learning Rate for Neural Networks

Tyler Moncur
Department of Mathematics, BYU
Master of Science

Neural networks have long been known as universal function approximators and have more recently been shown to be powerful and versatile in practice. But it can be extremely challenging to find the right set of parameters and hyperparameters. Model training is both expensive and difficult due to the large number of parameters and sensitivity to hyperparameters such as learning rate and architecture. Hyperparameter searches are notorious for requiring tremendous amounts of processing power and human resources.

This thesis provides an analytic approach to estimating the optimal value of one of the key hyperparameters in neural networks, the learning rate. Where possible, the analysis is computed exactly, and where necessary, approximations and assumptions are used and justified. The result is a method that estimates the optimal learning rate for a certain type of network, a fully connected CReLU network.

Keywords: neural network, learning rate, crelu

Contents

Contents **iii**

List of Figures **v**

1 Introduction **1**

1.1 Contributions 2

2 Background **2**

2.1 Neuron 2

2.2 Loss Notation 4

2.3 Looking Ahead 5

3 Crelu Networks **6**

3.1 Crelu Activation Function 6

3.2 Biases 9

3.3 CReLU Backpropagation 10

3.4 Loss Function 15

4 Hyperparameter Optimization **16**

4.1 Initialization 16

4.2 Quadratic Approximation 18

4.3 Spherical Symmetry 19

4.4 Orthogonality 21

4.5 First Derivative Expectation 25

4.6 Second Derivative Expectation 35

5 Testing **47**

5.1 First Step Testing 47

5.2 More Steps 49
5.3 Further Research 51
5.4 Conclusion 51

Bibliography **52**

List of Figures

2.1 Single Neuron Network 3
2.2 Hidden Layer Network 4

4.1 Theta Distributions Comparison 24
4.2 CIFAR100 Theta Distribution 24

5.1 Estimated vs True Optimal Learning Rates (Each dot is an estimated optimal learning rate. Each plus is a true optimal learning rate. Each color is a different network depth.) 49
5.2 Multiple Steps of Training (Each graph is a different architecture. Green line is using the optimal learning rate. Red line is using a larger learning rate. Blue line is using a smaller learning rate.) 50

Chapter 1. Introduction

The field of machine learning, especially neural networks, has made remarkable progress in recent years. Researchers continually surpass goals and previous record-holding benchmarks as we learn what is possible and how to more effectively build and train various models. This progress is steady and measurable, but this progress is often lacking a solid mathematical foundation. There is a growing effort to add rigor and understanding to the current methodology of building and training neural networks.

This problem was identified in a memorable talk by Ali Rahimi, where he stated that "machine learning has become alchemy". There is always a lot of guesswork that goes into choosing the many hyperparameters for a neural network such as the number of layers, learning rate, activation function and so on. While these decisions are guided by experience and trial and error, we would like to analyze these hyperparameters in more depth, and find mathematical foundations to help expedite the process of developing a neural network for solving a specific challenge.

Neural networks are both theoretically and practically very powerful function approximators. This is especially true in computer vision. Before 2012, the leading computer vision algorithms for object recognition generated feature vectors in a predefined fashion, like the SIFT algorithm. But the results from the 2012 ImageNet competition showed that a deep neural network, which can learn its own filters, could surpass these methods by a wide margin. This stunning result spurred on much of the current interest and research in the field, and it popularized many techniques that are still widely used in deep neural networks. This includes the ReLU activation function, which is so common that most deep learning practitioners use it without a second thought. And while it does work quite well, a more rigorous analysis on how to train a network is desirable.

My focus will be in analytically estimating the optimal learning rate for specific types of neural networks. Through a combination of direct calculation, and numerical approximation,

I attempt to estimate the optimal starting learning rate of a specific network at a lower cost than actual training. This work is in part an extension of the work by Chris Hettinger in this area. But we will diverge from his assumptions and results as we attempt to derive a more accurate estimation of the key values for training. Because of this, the final results do not agree with his conjectured formula.

1.1 Contributions

These results are primarily of theoretical value, but this is only because we limit the analysis to fully connected networks. My main contributions are:

- Using spherical symmetry to calculate the expectation of expressions involving the ReLU function and random matrices.
- Clearly identifying the assumptions made during the process of calculation.
- A new method of estimating the optimal learning rate of a particular type of neural network.

Chapter 2. Background

This section will briefly review the building blocks of a neural network and help develop the required notation. This section also presents a form of the expressions that must be evaluated later in this paper.

2.1 Neuron

The smallest component of a neural network is a single neuron. A neuron is composed of three parts, which are the linear transformation, the bias, and the non-linear activation function. We will use similar ideas and notation to Hettinger [1]. Let $\mathbf{w} \in \mathbb{R}^n$ be the weights

of the linear transformation, the bias $b \in \mathbb{R}$, and $a : \mathbb{R} \to \mathbb{R}$ the non-linear activation function. Then let $\mathbf{x} \in \mathbb{R}^n$ be the input to the neuron, and $\hat{y} \in \mathbb{R}$ the output.

$$\hat{y} = a(\mathbf{w}^T\mathbf{x} + b)$$

Each layer of a network is composed of one or more neurons. And in the case of a fully connected (dense) neural network, the input to all of the neurons in a given layer is the same. And the vector output of each layer feeds into the next layer. The number of neurons in a layer is the width.

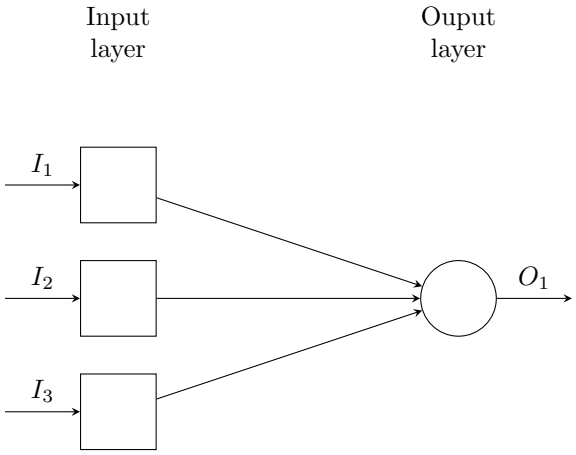

Figure 2.1: Single Neuron Network

Figure 2.1 above shows a visual representation of a single neuron, where the squares are the values of the input vector, and the circle is the only neuron. There are a variety of machine learning algorithms that are equivalent to a neural network with a single neuron and some preprocessing steps for the input. These single neuron networks work quite well on a range of problems, which may seem surprising because of how simple these methods are. This includes support vector machines and logistic regression, both of which have many variants. But the representative power of a model can be increased by stacking neurons together into larger networks. Consider the network in Figure 2.2.

This network has a what is called a hidden layer. This is the set of neurons that are

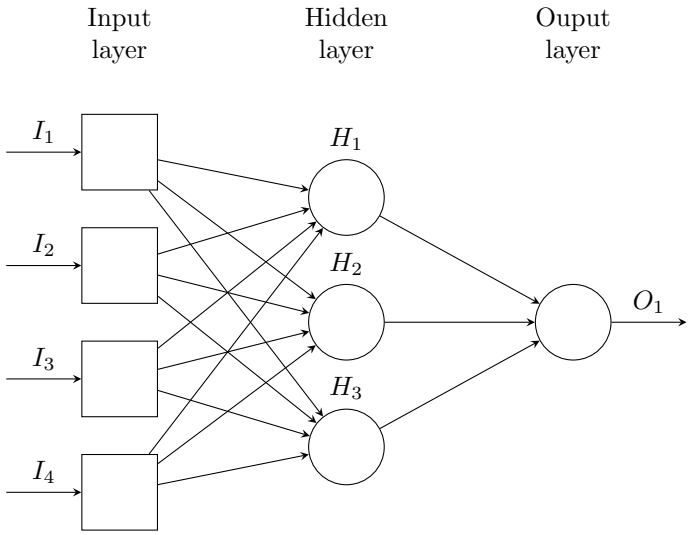

Figure 2.2: Hidden Layer Network

neither input nor output. This idea of a hidden layer is important to the representative power of a neural network, which we will return to while discussing the training of a network. The output from the neurons in this hidden layer is simply the input to the neurons in the final layer. The number of neurons contained in a layer is called its width, so in this case the hidden layer has a width of three neurons.

When building a network, only the input and output layers have a fixed size. These sizes are of course determined by the dimension of the input data values and target values in the dataset. Whereas the number of hidden layers and the size of each hidden layer is not fixed. This is part of what is called the architecture of a network, which includes the number and sizes of the layers used. The architecture is a key part of the hyperparameters that must be chosen when solving a specific problem.

2.2 Loss Notation

We will work with the loss function in many different forms throughout this paper, and the notation is slightly different based on which form is needed. Here I present the different notations and their definitions.

Notation Guide	
M($\mathbf{x}$, λ)	Network M applied to $\mathbf{x}$ after one step of gradient descent using λ.
$\mathbf{x}_i^k$	Output of layer i when the network is evaluated on the kth element of the dataset.
$L(\hat{\mathbf{y}}, \mathbf{y})$	The loss function applied to $\hat{\mathbf{y}}$ and $\mathbf{y}$.
$L(\mathbf{x}_n, \mathbf{y})$	Equivalent to the above.
$L(\mathbf{x}_n)$	Shorthand for the above.
$Ł_M$	The total loss of network M. $\left(\frac{1}{s}\sum_{k=1}^{s} L(\mathbf{x}_n^k)\right)$
$Ł_M(\lambda)$	Total loss after one step of gradient descent using λ. $\left(\frac{1}{s}\sum_{k=1}^{s} L(M(\mathbf{x}_0^k, \lambda))\right)$

Table 2.1: Data Features

2.3 Looking Ahead

As stated earlier, the goal of this paper is using analytical methods to estimate the optimal learning rate of a network. This learning rate is specific to a particular architecture and dataset. Once we get through all of the calculus and linear algebra, we find that most of the required expressions can be evaluated directly, though it is often tedious. There are two expressions, however, that are unlike the others, and they result from the activation function that is chosen. Here are simplified forms of the expressions that must be evaluated. Let $A \in \mathbb{R}^{p\times q}$ be a random matrix such that $a_{ij} \sim \mathcal{N}(0,1)$. Then the expressions of interest are the following.

$$\mathrm{E}\Big[\mathrm{ReLU}\Big((\mathbf{x}^T A^T \mathbf{z})(\mathbf{z}^T A\mathbf{y})\Big)\Big] \tag{2.1}$$

$$\mathrm{E}\Big[\mathrm{ReLU}\Big((\mathbf{x}^T A^T \mathbf{u})(\mathbf{u}^T A\mathbf{y})\Big)\mathrm{ReLU}\Big((\mathbf{x}^T A^T \mathbf{v})(\mathbf{v}^T A\mathbf{z})\Big)\Big] \tag{2.2}$$

These expressions prove difficult because ReLU is not linear, and therefore we cannot move the expectation inside of the ReLU function. Likewise, we cannot move the summations (resulting from the linear algebra) outside the ReLU function. We are forced to evaluate the expression without breaking it down into simpler terms. We find that (2.1) can be evaluated

exactly by using a trick involving spherical symmetry to reduce the dimension. In this case, it goes from $p \times q$ down to 2 dimensions. This calculation is shown in Lemma 4.11. The same trick can be applied to (2.2), however the results are still difficult to evaluate, and therefore, we assume many of the vectors are orthogonal in order to approximate the value. This assumption is explored in Lemma 4.3 and it is used to help estimate (2.2) in Lemma 4.19.

Chapter 3. CReLU Networks

In this section, we define the CReLU activation function and explain its practical and analytical advantages. We introduce more notation and calculate the derivatives needed for backpropagation in a CReLU network.

3.1 CReLU Activation Function

We have previously mentioned the need for a non-linear function when building a neural network. By far, the most widely used activation function for deep learning is the rectified linear unit, commonly referred to as the ReLU. It is defined as

$$\text{ReLU}(x) = \begin{cases} x & x > 0 \\ 0 & \text{otherwise.} \end{cases}$$

Here, the definition is written for a scalar x, but we often apply it element-wise to vectors. With that in mind, we will use the concatenated ReLU, or CReLU. This function is defined as

$$\text{CReLU}(\mathbf{x}) = \begin{bmatrix} \text{ReLU}(\mathbf{x}) \\ \text{ReLU}(-\mathbf{x}) \end{bmatrix}$$

Notice that this definition avoids the information loss that normally occurs with ReLU.

Rather than eliminating negative values, it separates the positive and negative components and treats them separately. The CReLU was developed multiple times with varied forms, names, and motivations. The name "concatenated ReLU" comes from Shang et. al.[2], who observed that the weights in AlexNet often developed opposite pairs. Published in the same year, Blot et. al. [3] developed an equivalent idea with the goal of maintaining information, and Kim et. al. [4] proposed using biologically inspired MaxMin maps to preserve both the positive and negative activations. And since then CReLU, has shown merit both for its simplification in analysis and its practical benefits.

In terms of practical benefits, we first need something to compare to. The ResNet is an excellent baseline. It does not fit the vanilla definition of a neural network, but its introduction in 2015 was nearly as dramatic as AlexNet was in 2012. The architecture allowed much deeper networks to train effectively, and the resulting networks won 1st place in several categories of the 2015 ImageNet Large Scale Visual Recognition Challenge [5]. With its success, a lot of work has gone into explaining why the architecture works. One explanation shows that making a ReLU network deeper turns the gradients into white noise where each value appears to be independent of its neighbors. This was termed the shattered gradients problem. In contrast, the ResNet produced gradients where neighboring activations were not independent. In 2017, Balzduzzi et. al. showed that CReLU compares favorably against the ResNet Architecture because it also solves the shattered gradients problem. But this practical benefit of CReLU is not the only reason to favor it. The CReLU function also simplifies the analysis of neural networks, especially when the network is initialized using the symmetric initialization, which will be discussed later.

Now we will show that CReLU networks have the same representative power as ReLU networks. To do this, we first review the statement of universality of neural networks.

Theorem 3.1. *(Hornik) Let $a : \mathbb{R} \to \mathbb{R}$ continuous and nonpolynomial. Let $X \subset \mathbb{R}^d$ be compact, and let $C(X)$ be the set of continuous function from X to R. Let F be the set of functions of the form*

$$\sum_{i=1}^{n} c_i a(\mathbf{w}_i^T \mathbf{x} + b_i)$$

where $\mathbf{w}_i \in \mathbb{R}^d$ *and* $b_i, c_i \in \mathbb{R}$. *The closure of* F *with respect to the uniform topology contains* $C(X)$.

Proof. See [6] □

Since the ReLU activation function is both continuous and nonpolynomial, Lemma 3.1 shows that neural networks with a hidden layer and ReLU are univeral function approximators. Or in other words, for any continuous $f : X \to \mathbb{R}$, there exists a single hidden layer network which approximates f with arbitrarily small error. Unfortunately, it is not a constructive proof, but rather it is only a proof of existence.

The proof does not immediately apply to a CReLU network as it does with ReLU. In the one dimensional case, we have $\text{CReLU} : \mathbb{R} \to \mathbb{R}^2$. Indeed, CReLU always doubles the dimensionality.

Proposition 3.2. *(Hettinger) The* CReLU *networks are universal function approximators.*

Proof. To show that CReLU networks are universal function approximators, we need only show that the collection of all ReLU networks form a subset of the collection of all CReLU networks. These two networks types are actually equivalent, but this is not required. Consider the single hidden layer network $M_{\text{ReLU}} : \mathbb{R}^d \to \mathbb{R}$ with the form

$$\hat{\mathbf{y}} = W_2 \text{ReLU}(W_1 \mathbf{x} + \mathbf{b}). \tag{3.1}$$

The ReLU network is discarding negative values; whereas, the CReLU network operates on the positive and negative values separately. To get the desired result, we must show that M_{ReLU} is equivalent to some CReLU network.

Let the CReLU network $M_{\text{CReLU}} : \mathbb{R}^d \to \mathbb{R}$ take the following form

$$\begin{aligned}
\hat{\mathbf{y}} &= \begin{bmatrix} P_2 & -N_2 \end{bmatrix} \mathrm{CReLU}(W_1'\mathbf{x} + \mathbf{b}) \\
&= \begin{bmatrix} P_2 & -N_2 \end{bmatrix} \begin{bmatrix} \mathrm{ReLU}(W_1'\mathbf{x} + \mathbf{b}) \\ \mathrm{ReLU}(-W_1'\mathbf{x} - \mathbf{b}) \end{bmatrix} \\
&= P_2\mathrm{ReLU}(W_1'\mathbf{x} + \mathbf{b}) - N_2\mathrm{ReLU}(-W_1'\mathbf{x} - \mathbf{b})
\end{aligned}$$

Choose $W_1' = W_1$, $P_2 = W_2$, and $N_2 = \mathbf{0}$. This makes this CReLU network equivalent to (3.1), and thus every ReLU layer can be converted into a CReLU layer. So ReLU networks form a subset of the collection of all CReLU networks and this implies that CReLU networks are also universal function approximators. □

3.2 Biases

For the sake of simplifying notation, we will use a common trick that hides the bias term inside the weight matrix. Consider the following expression

$$\mathbf{z} = W\mathbf{x} + \mathbf{b}$$

Now define

$$\mathbf{x}' = \begin{bmatrix} \mathbf{x} \\ 1 \end{bmatrix} \qquad W' = \begin{bmatrix} W & \mathbf{b} \\ \mathbf{0} & 1 \end{bmatrix} \qquad \mathbf{z}' = \begin{bmatrix} \mathbf{z} \\ 1 \end{bmatrix}$$

Note this trick requires an extra an dimension be added to the input vector. For a multi-layer network this only occurs with the initial input layer. The extra dimension is carried through to every following layer. The simplified expression is

$$\mathbf{z}' = W'\mathbf{x}'$$

From now on we will use this trick to avoid writing the bias and dealing with it separately.

3.3 CReLU Backpropagation

Much of the work done here is to simplify the notation for later on. We make use of the Heaviside function

$$h(x) = \begin{cases} 0 & x < 0 \\ \frac{1}{2} & x = 0 \\ 1 & x > 0 \end{cases}$$

Note that $h(x)$ only accepts scalar inputs. Several times, we have abused notation and allowed scalar functions to operate elementwise on vector inputs. That approach does not work in this case. Let $Hv(\mathbf{x})$ be defined as following.

$$Hv(\mathbf{x}) = \begin{bmatrix} h(x_1) & 0 & \dots & 0 \\ 0 & h(x_2) & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & h(x_n) \end{bmatrix} \tag{3.2}$$

Layer Notation. Let $\mathbf{x}_i$ be the output after evaluating the first i layers of a network.

$$\begin{aligned} \mathbf{x}_i &= \begin{bmatrix} P_i & -N_i \end{bmatrix} \mathrm{CReLU}(\mathbf{x}_{i-1}) \\ &= P_i \mathrm{ReLU}(\mathbf{x}_{i-1}) - N_i \mathrm{ReLU}(-\mathbf{x}_{i-1}) \\ &= P_i Hv(\mathbf{x}_{i-1})\mathbf{x}_{i-1} + N_i Hv(-\mathbf{x}_{i-1})\mathbf{x}_{i-1} \\ &= \big(P_i Hv(\mathbf{x}_{i-1}) + N_i Hv(-\mathbf{x}_{i-1})\big)\mathbf{x}_{i-1} \end{aligned}$$

Let $W_i = P_i Hv(\mathbf{x}_{i-1}) + N_i Hv(-\mathbf{x}_{i-1})$, thus $\mathbf{x}_i = W_i \mathbf{x}_{i-1}$. This form appears simpler, but

W_i is unfortunately defined in terms of $\mathbf{x}_{i-1}$. This brings us to the symmetric initialization. We will consider the initialization more carefully later, but the symmetric initialization is simply setting $P_i = N_i$ upon initialization of the network. This makes the network linear upon initialization, and makes analysis of the first step of gradient descent possible. Symmetric initialization also improves neural network training and performance [7]. For the remainder of the paper, we will assume a symmetric initialization. When $P_i = N_i$, we get the following simplification.

$$\begin{aligned} W_i &= (P_i Hv(\mathbf{x}_{i-1}) + N_i Hv(-\mathbf{x}_{i-1})) \\ &= P_i(Hv(\mathbf{x}_{i-1}) + Hv(-\mathbf{x}_{i-1})) \\ &= P_i \end{aligned}$$

The last step follows from $h(x) + h(-x) = 1 \implies Hv(\mathbf{x}) + Hv(-\mathbf{x}) = I$. Thus W_i is independent of $\mathbf{x}_{i-1}$ when evaluating the network on initialization. We use the following notation for convenience.

Definition 3.3. For $a \geq b$, let

$$W_{a,b} = W_a W_{a-1} \cdots W_b$$

And for $a < b$, let $W_{a,b} = 1$.

With this new notation we have $\mathbf{x}_i = W_{i,1}\mathbf{x}_0$. To calculate the gradients of the network, we apply the chain rule repeatedly. Therefore, we need all of the intermediate derivatives.

Lemma 3.4. *The following relations hold:*

$$\frac{\partial}{\partial P_i} L(\hat{\mathbf{y}}, \mathbf{y}) = \frac{\partial}{\partial \mathbf{x}_i} L(\hat{\mathbf{y}}, \mathbf{y}) \mathrm{ReLU}(\mathbf{x}_{i-1})^T \qquad = \frac{\partial}{\partial \mathbf{x}_i} L(\hat{\mathbf{y}}, \mathbf{y}) \mathbf{x}_{i-1}^T Hv(\mathbf{x}_{i-1}) \tag{3.3}$$

$$\frac{\partial}{\partial N_i} L(\hat{\mathbf{y}}, \mathbf{y}) = -\frac{\partial}{\partial \mathbf{x}_i} L(\hat{\mathbf{y}}, \mathbf{y}) \mathrm{ReLU}(-\mathbf{x}_{i-1})^T = \frac{\partial}{\partial \mathbf{x}_i} L(\hat{\mathbf{y}}, \mathbf{y}) \mathbf{x}_{i-1}^T Hv(-\mathbf{x}_{i-1}) \tag{3.4}$$

$$\frac{\partial}{\partial \mathbf{x}_i} L(\hat{\mathbf{y}}, \mathbf{y}) = (W_{i+1})^T \frac{\partial}{\partial \mathbf{x}_{i+1}} L(\hat{\mathbf{y}}, \mathbf{y}) \qquad = (P_{i+1} Hv(\mathbf{x}_i) + N_{i+1} Hv(-\mathbf{x}_i))^T \frac{\partial}{\partial \mathbf{x}_{i+1}} L(\hat{\mathbf{y}}, \mathbf{y}). \tag{3.5}$$

Proof. Note the following

$$\mathbf{x}_i = P_i \mathrm{ReLU}(\mathbf{x}_{i-1}) - N_i \mathrm{ReLU}(-\mathbf{x}_{i-1}) \tag{3.6}$$

$$\mathbf{x}_{i+1} = W_{i+1} \mathbf{x}_i \tag{3.7}$$

We start with $\frac{\partial}{\partial P_i} L(\hat{\mathbf{y}}, \mathbf{y})$. Note that this expression is a matrix with the same dimension as P_i. We perform the computation on a single element at a time. Note that

$$\left(\frac{\partial}{\partial P_i} L(\hat{\mathbf{y}}, \mathbf{y})\right)_{jk} = \frac{\partial}{\partial P_{ijk}} L(\hat{\mathbf{y}}, \mathbf{y})$$
$$\left(\frac{\partial}{\partial \mathbf{x}_i} L(\hat{\mathbf{y}}, \mathbf{y})\right)_{j} = \frac{\partial}{\partial \mathbf{x}_{ij}} L(\hat{\mathbf{y}}, \mathbf{y})$$

We have

$$\begin{aligned}
\left(\frac{\partial}{\partial P_i} L(\hat{\mathbf{y}}, \mathbf{y})\right)_{jk} &= \frac{\partial}{\partial P_{ijk}} L(\hat{\mathbf{y}}, \mathbf{y}) \\
&= \frac{\partial}{\partial \mathbf{x}_{ij}} L(\hat{\mathbf{y}}, \mathbf{y}) \frac{\partial}{\partial P_{ijk}} \mathbf{x}_{ij} && \text{Chain rule} \\
&= \frac{\partial}{\partial \mathbf{x}_{ij}} L(\hat{\mathbf{y}}, \mathbf{y}) \mathrm{ReLU}(\mathbf{x}_{i-1,k}) && \text{Derivative of (3.6)} \\
&= \left(\frac{\partial}{\partial \mathbf{x}_i} L(\hat{\mathbf{y}}, \mathbf{y})\right)_j \left(\mathrm{ReLU}(\mathbf{x}_{i-1})\right)_k
\end{aligned}$$

Note that for the outer product $\mathbf{u}\mathbf{v}^T$, we have $(\mathbf{u}\mathbf{v}^T)_{jk} = \mathbf{u}_j \mathbf{v}_k$. Also note $\mathrm{ReLU}(\mathbf{x}) = Hv(\mathbf{x})\mathbf{x}$. Thus we have

$$\begin{aligned}
\left(\tfrac{\partial}{\partial P_i} L(\hat{\mathbf{y}}, \mathbf{y})\right)_{jk} &= \left(\tfrac{\partial}{\partial \mathbf{x}_i} L(\hat{\mathbf{y}}, \mathbf{y})\right)_j \left(\mathrm{ReLU}(\mathbf{x}_{i-1})\right)_k \\
\left(\tfrac{\partial}{\partial P_i} L(\hat{\mathbf{y}}, \mathbf{y})\right)_{jk} &= \left(\tfrac{\partial}{\partial \mathbf{x}_i} L(\hat{\mathbf{y}}, \mathbf{y}) \mathrm{ReLU}(\mathbf{x}_{i-1})^T\right)_{jk} \\
\tfrac{\partial}{\partial P_i} L(\hat{\mathbf{y}}, \mathbf{y}) &= \tfrac{\partial}{\partial \mathbf{x}_i} L(\hat{\mathbf{y}}, \mathbf{y}) \mathrm{ReLU}(\mathbf{x}_{i-1})^T = \tfrac{\partial}{\partial \mathbf{x}_i} L(\hat{\mathbf{y}}, \mathbf{y}) \mathbf{x}_{i-1}^T Hv(\mathbf{x}_{i-1})
\end{aligned}$$

Similar manipulations give the second result

$$\tfrac{\partial}{\partial N_i} L(\hat{\mathbf{y}}, \mathbf{y}) = -\tfrac{\partial}{\partial \mathbf{x}_i} L(\hat{\mathbf{y}}, \mathbf{y}) \mathrm{ReLU}(-\mathbf{x}_{i-1})^T = \tfrac{\partial}{\partial \mathbf{x}_i} L(\hat{\mathbf{y}}, \mathbf{y}) \mathbf{x}_{i-1}^T Hv(-\mathbf{x}_{i-1}).$$

For the third result, note that $\mathbf{x}_{i+1}$ depends on all elements of $\mathbf{x}_i$, which gives us the following when applying the chain rule:

$$\begin{aligned}
\left(\tfrac{\partial}{\partial \mathbf{x}_i} L(\hat{\mathbf{y}}, \mathbf{y})\right)_j &= \tfrac{\partial}{\partial \mathbf{x}_{ij}} L(\hat{\mathbf{y}}, \mathbf{y}) \\
&= \sum_{k=1}^{n} \tfrac{\partial}{\partial \mathbf{x}_{i+1,k}} L(\hat{\mathbf{y}}, \mathbf{y}) \tfrac{\partial}{\partial \mathbf{x}_{ij}} \mathbf{x}_{i+1,k} && \text{Chain rule} \\
&= \sum_{k=1}^{n} \tfrac{\partial}{\partial \mathbf{x}_{i+1,k}} L(\hat{\mathbf{y}}, \mathbf{y}) W_{i+1,k,j} && \text{Derivative of (3.7)} \\
&= \sum_{k=1}^{n} (\tfrac{\partial}{\partial \mathbf{x}_{i+1}} L(\hat{\mathbf{y}}, \mathbf{y}))_k (W_{i+1}^T)_{jk}
\end{aligned}$$

Note that $(A\mathbf{v})_j = \sum_{k=1}^{n} A_{jk} \mathbf{v}_k$. Thus we have

$$\begin{aligned}
\left(\tfrac{\partial}{\partial \mathbf{x}_i} L(\hat{\mathbf{y}}, \mathbf{y})\right)_j &= \sum_{k=1}^{n} (\tfrac{\partial}{\partial \mathbf{x}_{i+1}} L(\hat{\mathbf{y}}, \mathbf{y}))_k (W_{i+1}^T)_{jk} \\
\left(\tfrac{\partial}{\partial \mathbf{x}_i} L(\hat{\mathbf{y}}, \mathbf{y})\right)_j &= (W_{i+1}^T \tfrac{\partial}{\partial \mathbf{x}_{i+1}} L(\hat{\mathbf{y}}, \mathbf{y}))_j \\
\tfrac{\partial}{\partial \mathbf{x}_i} L(\hat{\mathbf{y}}, \mathbf{y}) &= W_{i+1}^T \tfrac{\partial}{\partial \mathbf{x}_{i+1}} L(\hat{\mathbf{y}}, \mathbf{y}) = (P_{i+1} Hv(\mathbf{x}_i) + N_{i+1} Hv(-\mathbf{x}_i))^T \tfrac{\partial}{\partial \mathbf{x}_{i+1}} L(\hat{\mathbf{y}}, \mathbf{y}) \qquad \square
\end{aligned}$$

Recall that a neural network is updated using the gradient according to the rule

$$W \leftarrow W - \lambda \frac{\partial L_M}{\partial W}$$

Since we are dealing with a dataset of inputs, we now will use superscript notation $\mathbf{x}_0^k$ to denote the kth element of the dataset. Also let $\mathbf{x}_i^k$ and $\mathbf{y}^k$ be corresponding outputs from $\mathbf{x}_0^k$. The total loss is $L_M = \frac{1}{s}\sum_{k=1}^{s} L(\hat{\mathbf{y}}^k, \mathbf{y}^k)$, so the total gradients take the form

$$\frac{\partial L_M}{\partial P_i} = \frac{1}{s}\sum_{k=1}^{s} \frac{\partial}{\partial P_i} L(\hat{\mathbf{y}}^k, \mathbf{y}^k)$$

and

$$\frac{\partial L_M}{\partial N_i} = \frac{1}{s}\sum_{k=1}^{s} \frac{\partial}{\partial N_i} L(\hat{\mathbf{y}}^k, \mathbf{y}^k).$$

A full update step will be

$$P_i \leftarrow P_i - \lambda \frac{\partial L_M}{\partial P_i}$$
$$N_i \leftarrow N_i - \lambda \frac{\partial L_M}{\partial N_i}.$$

Now, we make one more simplification to the notation. We will no longer use $\hat{\mathbf{y}}$ to denote the output of the network. Instead we use $\mathbf{x}_n$, since it is also the output of the nth layer. We also shorten the notation for the loss.

Definition 3.5. We write

$$L(\mathbf{x}_n^k) = L(\mathbf{x}_n^k, \mathbf{y}^k).$$

We need this shorthand to make some of the later formulas fit on the page.

3.4 Loss Function

The analysis does not require a specific loss function. It only requires that the loss function be twice differentiable. Presented below are two of the most common loss functions.

Cross Entropy. Cross entropy is generally used for classification problems. Let n be the number of classes.

Define softmax : $\mathbb{R}^n \to \mathbb{R}^n$ as

$$\text{softmax}(\mathbf{x}) = \frac{1}{e^{x_1} + \cdots + e^{x_n}} \begin{bmatrix} e^{x_1} \\ \vdots \\ e^{x_n} \end{bmatrix}.$$

Define CrossEntropy : $\mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}^n$ as

$$\text{CrossEntropy}(\mathbf{y}, \hat{\mathbf{y}}) = -\sum_{i=1}^{n} y_i \log(\hat{y}_i).$$

With classification problems, the target for each sample is the correct class, which is represented by a vector of zeros except for a single one at the index corresponding to the correct class. This is called one-hot encoding. It also requires that the output of network represent a probability, hence it is often preceded by a softmax layer, which converts the outputs to positive values which sum to one.

Mean Squared Error. Mean Squared Error is used for regression problems where a real valued number is the target.

Define MSE : $\mathbb{R}^n \to \mathbb{R}$ as

$$\text{MSE}(\mathbf{y}, \hat{\mathbf{y}}) = \frac{1}{n} \sum_{i=1}^{n} (y_i - \hat{y}_i)^2 = \frac{1}{n} \left\| \mathbf{y} - \hat{\mathbf{y}} \right\|_2^2$$

One challenge with loss functions used in regression is dealing with outliers. The Mean Squared Error Loss is particularly affected by outlier target values, and this should be considered when choosing it as a loss function.

Note that both loss functions presented are twice differentiable. This is essential because we rely on the gradient of the loss to train the weights of a network. Fortunately, the analysis does not require a particular loss function. All we require is that it be twice differentiable.

Chapter 4. Hyperparameter Optimization

In this section, we show how to approximate the optimal learning rate, in terms of derivatives of the total loss $L_M(\lambda)$. We define the optimal learning rate, $\lambda_{optimal}$, as the learning rate which minimizes the total loss of the function after one step of gradient descent. We show in Section 4.2 that Newton's method gives the following approximation to $\lambda_{optimal}$.

$$\lambda_{optimal} \approx -\frac{\frac{d}{d\lambda}L_M(0)}{\frac{d^2}{d\lambda^2}L_M(0)}$$

.

Then we evaluate the expectations of the numerator and denominator over the random initialization. In Theorem 4.13 we show

$$\mathrm{E}\left[\frac{d}{d\lambda}L_M(0)\right] \approx -\frac{1}{s^2}\sum_{i=1}^{n}\frac{\sqrt{d_i d_{i-1}}}{\sqrt{d_0 d_n}}\sum_{k=1}^{s}\sum_{l=1}^{s}\frac{\partial}{\partial \mathbf{x}_n^l}L(\mathbf{x}_n^l)^T\frac{\partial}{\partial \mathbf{x}_n^k}L(\mathbf{x}_n^k)\frac{\left\|\mathbf{x}_0^k\right\|\left\|\mathbf{x}_0^l\right\|}{\pi}((\pi-\theta_{kl})\cos\theta_{kl}+\sin\theta_{kl})$$

And $\mathrm{E}\left[\frac{d^2}{d\lambda^2}L_M(0)\right]$ is available in Theorem 4.24.

4.1 Initialization

A good set of initial weights is important to training a neural network. I will present the method shown by Hettinger and compare it to other common initializations. But first, let's review simpler, though less useful, initializations.

In terms of notation, we use w_{ijk} to refer to the element of W_i at the jth row and kth column. The size of the forward signal at a given layer refers to magnitude of the $\mathbf{x}_i$, and

the backward signal refers to the magnitude of $\frac{d}{d\mathbf{x}_i}L(\mathbf{x}_i)$.

Zero Initialization. Using all zeros ($w_{ijk} = 0$) is generally not recommended because it brings symmetry into a nonconvex optimization problem. With neural network optimization, saddle points are points where the gradient is nearly zero in all directions, but it is not a local minimum. Randomness can be helpful in avoiding and leaving saddle points, whereas symmetry can be problematic.

Normal Initialization. Letting $w_{ijk} \sim \mathcal{N}(0, 1)$ seems like a reasonable solution to the saddle points, and it works well in small networks. However, in deep networks, if the layers are not scaled properly, they can cause either exponential growth or decay in the forward and backward signals. This makes gradient descent unstable.

Scaled Normal Initializations. The following methods of initialization are each a version of a properly scaled normal initialization. Consider a layer with weight matrix W_i. Let W_i have dimensions $d_i \times d_{i-1}$, so the input dimension of this layer is d_{i-1}.

The Xavier Initialization is $w_{jk} \sim \mathcal{N}(0, 1/d_{i-1})$. This initialization is created for a neural network using a particular activation function, the *tanh* function.

The He Initialization is $w_{ijk} \sim \mathcal{N}(0, 2/d_{i-1})$. This initialization is created for a neural network using the ReLU activation function. The factor of 2 is used because approximately half of incoming dimensions are zero thanks to ReLU.

Both of these initialization schemes were created to manage the growth of signals in the forward pass of their respective network type, thus preventing exponential growth or decay. Similar results are achieved by managing the signals of the backward pass [8].

For a CReLU network, we use the Hettinger initialization, which attempts to balance the signal growth of both the forward and backward pass. Hettinger uses the following

$$w_{ijk} \sim \mathcal{N}(0, \tfrac{1}{\sqrt{d_{i-1}d_i}})$$

$$P_i = N_i.$$

As shown by Hettinger in Theorem 4.6 [1], this choice of initialization gives the following property

$$\frac{\mathrm{E}\left[\left\|\frac{d}{dP_i}L(\mathbf{x}_n)\right\|^2\right]}{\mathrm{E}\left[\left\|\frac{d}{dP_j}L(\mathbf{x}_n)\right\|^2\right]} = \frac{\mathrm{E}\left[\|P_i\|^2\right]}{\mathrm{E}\left[\|P_j\|^2\right]} \qquad \frac{\mathrm{E}\left[\left\|\frac{d}{dN_i}L(\mathbf{x}_n)\right\|^2\right]}{\mathrm{E}\left[\left\|\frac{d}{dN_j}L(\mathbf{x}_n)\right\|^2\right]} = \frac{\mathrm{E}\left[\|N_i\|^2\right]}{\mathrm{E}\left[\|N_j\|^2\right]}$$

This property shows the balance between the forward and backward signals. And when the full network is considered, one can show that the magnitude of the forward signal (and hence also the backward signal) is independent of the network depth and width. We use the Hettinger initialization in the analysis.

4.2 Quadratic Approximation

Let's define the function we want to optimize. We intend to minimize the loss of the network by choosing the best learning rate. To make an analysis possible, we will search for the learning rate that minimizes the loss after a single step of gradient descent. Therefore we need a function that describes the loss after the first step in terms of a given learning rate. Since we already use N for the negative activation matrix, let M be a network. Let $M(\mathbf{x}_0^k, \lambda)$ return the output for $\mathbf{x}_0^k$ after optimizing M with one step of gradient descent with learning rate λ. Note that $M(\mathbf{x}_0^k, 0)$ gives the output of the initial network. Denote the total loss after one step as

$$L_M(\lambda) = \tfrac{1}{s}\sum_{k=1}^{s} L(M(\mathbf{x}_0^k, \lambda), \mathbf{y}^k).$$

Neural networks are nonconvex and nonpolynomial, which makes optimization significantly more challenging. In this case, Newton's method is a reasonable approach to minimizing the total loss function with respect to λ. Newton's method is generally considered a powerful optimizer since it minimizes a quadratic approximation rather than a linear one. It

is not used for neural network optimization because the computational space scales as $O(n^2)$ with the dimension of the problem, and most neural networks have millions of parameters. But this minimization problem is 1-dimensional since it is only with respect to λ. Newton's method of optimization is iterative, using the following formula:

$$x_{i+1} = x_i - \frac{f'(x_i)}{f''(x_i)}.$$

We require an initial guess of $\lambda = 0$ since this is equivalent to starting out with the initial network. Thus we have the following approximation for $\lambda_{optimal}$.

$$\lambda_{optimal} \approx -\frac{\frac{d}{d\lambda}L_M(0)}{\frac{d^2}{d\lambda^2}L_M(0)}$$

This equation is made more useful by focusing on the expectation of these expressions. Since the initialization is randomized, we can take the expectation $\mathrm{E}\big[\frac{d}{d\lambda}L_M(0)\big]$ over the possible initial weight values.

Thus, we must now calculate or estimate the values $\mathrm{E}\big[\frac{d}{d\lambda}L_M(0)\big]$ and $\mathrm{E}\Big[\frac{d^2}{d\lambda^2}L_M(0)\Big]$. The next two sections produce important tools for calculating these expectations.

4.3 Spherical Symmetry

Lemma 4.1. *Let $h : \mathbb{R}^{r\times s} \to \mathbb{R}^{r\times s}$ be invertible, differentiable, and volume preserving. Let X be a random variable with dimension $r \times s$ and probability density function $f_X(x)$. Let $Y = h(X)$ have probability density function $f_Y(y)$. If $f_X(x) = f_X(h^{-1}(x))$ for all x, then the distributions of X and Y are identical.*

Proof. We must show $f_X(x) = f_Y(x)$. Note that

$$Y = h(X) \implies f_Y(y) = f_X(h^{-1}(y)) \left|\det\Big(\frac{d}{dy}h^{-1}(y)\Big)\right|.$$

Since $h(x)$ is volume preserving, this simplifies to $f_Y(y) = f_X(h^{-1}(y))$. By the hypothesis,

$f_X(x) = f_X(h^{-1}(x))$, therefore $f_X(x) = f_X(h^{-1}(x)) = f_Y(x)$. □

Lemma 4.2. *Let $A \in \mathbb{R}^{r\times s}$ be a random matrix such that $a_{ij} \sim \mathcal{N}(0, \sigma^2)$, and all a_{ij} are independent. Let $P \in \mathbb{R}^{r\times r}$ and $Q \in \mathbb{R}^{s\times s}$ be orthonormal matrices. Then A and PAQ are identically distributed.*

Proof. Let $f : \mathbb{R}^{r\times s} \to \mathbb{R}$ be the probability density function for A. Define $h(A) = PAQ$. Since $h(A)$ only involves multiplying by rotational matrices, it is volume preserving. Then by Lemma 4.1, we need only show that $f(A) = f(h^{-1}(A))$. The inverse function is simply $h^{-1}(A) = P^T AQ^T$. This is clearly seen by

$$h^{-1}(h(A)) = h^{-1}(PAQ) = P^T PAQQ^T = A$$

$$h(h^{-1}(A)) = h(P^T AQ^T) = PP^T AQ^T Q = A.$$

This is true because P and Q are orthonormal which implies $P^T P = PP^T = I$ and $Q^T Q = QQ^T = I$.

Now we find $f(A)$. Because the entries of A are independent, the joint probability distribution is simply a product of the probability distribution functions of each individual entry.

$$f(A) = \prod_{i=1}^{r}\prod_{j=1}^{s} \tfrac{1}{\sqrt{2\pi\sigma^2}} \exp(-\tfrac{a_{ij}^2}{2\sigma^2}) \tag{4.1}$$

$$= (2\pi\sigma^2)^{-\frac{rs}{2}} \exp(-\sum_{i=1}^{r}\sum_{j=1}^{s} \tfrac{a_{ij}^2}{2\sigma^2}) \tag{4.2}$$

$$= (2\pi\sigma^2)^{-\frac{rs}{2}} \exp(-\tfrac{\|A\|_F^2}{2\sigma^2}) \tag{4.3}$$

So $f(A)$ depends only on the square of the Frobenius norm of A. Therefore, we need only show $\|A\|_F^2 = \|h^{-1}(A)\|_F^2$. The square of the Frobenius norm is given by $\|A\|_F^2 = \mathrm{Tr}(A^T A)$.

Note the cyclic property of the trace gives $\mathrm{Tr}(ABC) = \mathrm{Tr}(BCA)$. Thus

$$\begin{aligned}
\left\|h^{-1}(A)\right\|_F^2 &= \left\|P^T A Q^T\right\|_F^2 \\
&= \mathrm{Tr}(Q A^T P P^T A Q^T) \\
&= \mathrm{Tr}(A^T P P^T A Q^T Q) \\
&= \mathrm{Tr}(A^T A) \\
&= \|A\|_F^2
\end{aligned}$$

□

Another interpretation of this result is that the distribution of the matrix A is spherically symmetric. This is true because an orthonormal matrix is also a rotational matrix. Since each weight matrix in a CReLU network is initialized with normal, independent values, we can immediately apply this lemma with any of the weight matrices in the network. This lemma is most useful for reducing the dimensions to simplify an expectation. We have few other options for simplifying the expectations when they involve the ReLU function. But because $\mathrm{ReLU}(r \cdot x) = r \cdot \mathrm{ReLU}(x)$ for $r \geq 0$, we can always factor out the radius after converting to spherical coordinates. This lemma is used for computations in Lemma 4.11, Lemma 4.19, and Lemma 4.20.

4.4 Orthogonality

One of the key approximations needed for later calculations is that large random vectors tend to be nearly orthogonal to each other. The previous statement is vague. The following lemma provides a rigorous statement on the matter to clarify when it can be assumed that two random vectors are nearly orthogonal. Further treatment of the subject can be found in [9].

Lemma 4.3. *Let $\mathbf{x}, \mathbf{y} \in \mathbb{R}^n$ be vectors of random variables whose elements are mutually independent and normally distributed as $\mathcal{N}(0,1)$. Let $\theta = \cos^{-1}(\frac{\mathbf{x}^T\mathbf{y}}{\|\mathbf{x}\|\|\mathbf{y}\|})$. Then we have the*

following for the probability density function of θ.

$$f_\theta(\theta_0) \propto \sin^{n-2}\theta_0$$

Proof. Here is a sketch of the proof. We by start by calculating the cumulative density function. So we want to calculate $P(\theta \leq \theta_0)$. First, we make several simplifications. Note that the magnitudes of $\mathbf{x}$ and $\mathbf{y}$ do not influence θ, so we can replace them with $\hat{\mathbf{x}} = \frac{\mathbf{x}}{\|\mathbf{x}\|}$ and $\hat{\mathbf{y}} = \frac{\mathbf{y}}{\|\mathbf{y}\|}$. Since $\mathbf{x}$ and $\mathbf{y}$ were both spherically symmetric, we know that $\hat{\mathbf{x}}$ and $\hat{\mathbf{y}}$ are uniformly distributed on the surface of a unit n-sphere. Because of this symmetry and since we only care about the angle between $\hat{\mathbf{x}}$ and $\hat{\mathbf{y}}$, we can proceed assuming $\hat{\mathbf{x}} = \mathbf{e}_1$ without loss of generality.

First note that $\hat{\mathbf{x}}^T\hat{\mathbf{y}} = \cos\theta$. Since we are restricted to $0 \leq \theta \leq \pi$, we have that $\theta \leq \theta_0 \iff \hat{\mathbf{x}}^T\hat{\mathbf{y}} \geq \cos\theta_0$. Thus we want $P(\hat{\mathbf{x}}^T\hat{\mathbf{y}} \geq \cos\theta_0)$. Since we can assume $\hat{\mathbf{x}} = \mathbf{e}_1$, we have

$$\begin{aligned}
P(\theta \leq \theta_0) &= P(\hat{\mathbf{x}}^T\hat{\mathbf{y}} \geq \cos\theta_0) \\
&= P(\mathbf{e}_1^T\hat{\mathbf{y}} \geq \cos\theta_0) \\
&= P(\hat{y}_1 \geq \cos\theta_0) \\
&= \int_{\{\hat{\mathbf{y}} \in S^n | \hat{y}_1 \geq \cos\theta_0\}} dP \\
&= \int_{\{\hat{\mathbf{y}} \in S^n | \hat{y}_1 \geq \cos\theta_0\}} f_{\hat{\mathbf{y}}}(\hat{\mathbf{y}}) dy_1 dy_2 \cdots d_{y_n}
\end{aligned}$$

Here, S^n is the surface of the unit n-sphere. Fortunately, since $\hat{\mathbf{y}}$ is uniform, we know $f_{\hat{\mathbf{y}}}(\hat{\mathbf{y}})$ is constant. Thus the integral above is simply a constant times the hyper-volume of the region $\{\hat{\mathbf{y}} \in S^n | \hat{y}_1 \geq \cos\theta_0\}$. Let's switch to hyperspherical coordinates.

$$\begin{aligned}\int_{\{\hat{\mathbf{y}}\in S^n|\hat{y}_1\geq\cos\theta_0\}} f(\hat{\mathbf{y}})dy_1\cdots d_{y_n} &\propto \int_{\{\hat{\mathbf{y}}\in S^n|\hat{y}_1\geq\cos\theta_0\}} dy_1\cdots d_{y_n}\\ &= \int_{\{\hat{\mathbf{y}}\in S^n|\phi_1\leq\theta_0\}} \sin^{n-2}\phi_1\cdots\sin\phi_{n-2}d\phi_1\cdots d_{\phi n-1}\\ &= \int_0^{\theta_0}\sin^{n-2}\phi_1 d\phi_1\int_{\{\hat{\mathbf{y}}\in S^n\}}\sin^{n-3}\phi_2\cdots\sin\phi_{n-3}\phi_2 d\phi_2\cdots d_{\phi n-1}\\ &\propto \int_0^{\theta_0}\sin^{n-2}\phi_1 d\phi_1\end{aligned}$$

In the last line, we drop the second integral since it is constant with respect to θ_0. Thus we have shown

$$P(\theta\leq\theta_0)\propto\int_0^{\theta_0}\sin^{n-2}\phi_1 d\phi_1$$

But since we are interested in the probability density function of θ, we take the derivative of both sides with respect to θ_0.

$$\begin{aligned}\frac{\partial}{\partial\theta_0}P(\theta\leq\theta_0)&\propto\frac{\partial}{\partial\theta_0}\int_0^{\theta_0}\sin^{n-2}\phi_1 d\phi_1\\ f_\theta(\theta_0)&\propto\sin^{n-2}\theta_0\end{aligned}$$

□

Now let's consider the meaning of this result. As n increases, the sine curve becomes more peaked around the value $\frac{\pi}{2}$. Thus it becomes increasingly likely for $\mathbf{x}$ and $\mathbf{y}$ be nearly orthogonal.

Figure 4.1 shows the probablity density function for θ for various dimensions. In its scaled form the pdf is $f(\theta_0)=\frac{1}{\sqrt{\pi}}\frac{\Gamma(\frac{n}{2})}{\Gamma(\frac{n-1}{2})}\sin^{n-2}(\theta_0)$. While this is interesting, note that we are not working with datapoints that are distributed in a uniform spherically symmetric pattern. Just because the datapoints of CIFAR100 each have 3072 dimensions does not mean we can state they are all almost surely pairwise nearly orthogonal. Usually, the data we are concerned with falls on a manifold of much lower dimension. For example, through

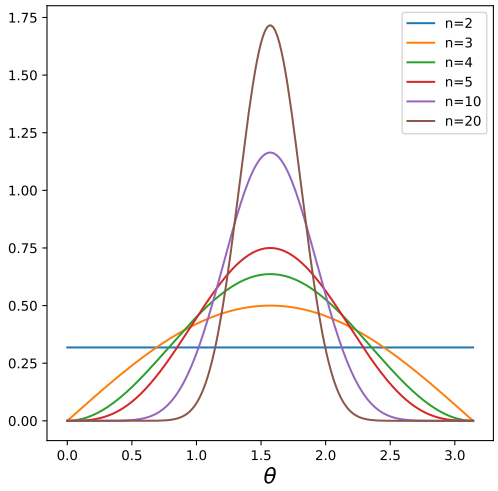

Figure 4.1: Theta Distributions Comparison

numerical experiment, I found the distribution of angles in the CIFAR100 dataset was fit quite well by the curve above when $n = 14.9$. This suggests the dataset is approximately 15 dimensional. The comparison is shown in Figure 4.2.

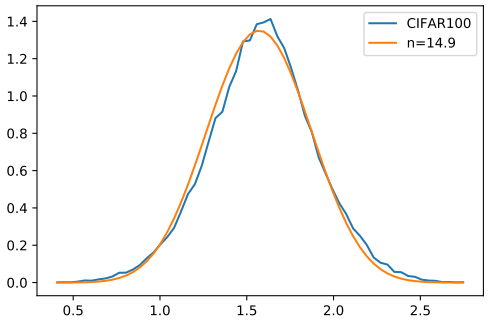

Figure 4.2: CIFAR100 Theta Distribution

Therefore, we should be cautious when we assume that large vectors are nearly orthogonal. Generally, we would desire more information before making this assumption. Despite this, we make this assumption in Lemma 4.19, Lemma 4.20, and Lemma 4.21. And the approximation works well enough as we will see in the testing.

4.5 First Derivative Expectation

The following helps us by simplifying an expression that appears in the calculation of both the first and second derivatives.

Lemma 4.4. *For* $\mathbf{x}, \mathbf{y} \in \mathbb{R}^n$ *we have*

$$\sum_{i=1}^{n} \mathrm{ReLU}(x_i y_i) = \mathbf{x}^T (Hv(\mathbf{x})Hv(\mathbf{y}) + Hv(-\mathbf{x})Hv(-\mathbf{y}))\mathbf{y}$$

where $Hv(\mathbf{x})$ *was defined in* (3.2).

Proof. Note that

$$\mathbf{x}^T (Hv(\mathbf{x})Hv(\mathbf{y}) + Hv(-\mathbf{x})Hv(-\mathbf{y}))\mathbf{y} = \sum_{i=1}^{n} x_i (h(x_i)h(y_i) + h(-x_i)h(-y_i))y_i.$$

Thus it is sufficient to show that $\mathrm{ReLU}(x_i y_i) = x_i(h(x_i)h(y_i)+h(-x_i)h(-y_i))y_i$. By symmetry we need only consider the following cases.

Case $x_i, y_i > 0$:

$$\mathrm{ReLU}(x_i y_i) = x_i y_i$$

$$x_i(h(x_i)h(y_i) + h(-x_i)h(-y_i))y_i = x_i(1+0)y_i = x_i y_i$$

Case $x_i > 0, y_i < 0$:

$$\mathrm{ReLU}(x_i y_i) = 0$$

$$x_i(h(x_i)h(y_i) + h(-x_i)h(-y_i))y_i = x_i(0+0)y_i = 0$$

Case $x_i, y_i < 0$:

$$\mathrm{ReLU}(x_i y_i) = x_i y_i$$

$$x_i(h(x_i)h(y_i) + h(-x_i)h(-y_i))y_i = x_i(0+1)y_i = x_i y_i$$

Case $x_i = 0$ or $y_i = 0$: Then clearly both expressions go to 0. □

Definition 4.5. (Modified from Hettinger) Recall that d_i is the number of rows in W_i. We define the following scalar values, which appear repeatedly.

$$\alpha_{ikl} = \sum_{m=1}^{d_i} \mathrm{ReLU}\big((\mathbf{x}_0^l)^T (W_{i-1,1})^T \mathbf{e}_m \mathbf{e}_m^T W_{i-1,1} \mathbf{x}_0^k\big) \tag{4.4}$$

$$\omega_{ikl} = \tfrac{\partial}{\partial \mathbf{x}_n^k} L(\mathbf{x}_n^k)^T W_{n,i+1} (W_{n,i+1})^T \tfrac{\partial}{\partial \mathbf{x}_n^l} L(\mathbf{x}_n^l)^T \tag{4.5}$$

The scalars defined above are chosen because they form the most natural way to break up the computation of the first derivative. Now we derive an expression for the first derivative.

Lemma 4.6. *(Modified from Hettinger)*

$$\frac{d}{d\lambda} L_M(\lambda)\big|_{\lambda=0} = -\tfrac{1}{s^2} \sum_{k=1}^{s} \sum_{i=1}^{n} \sum_{l=1}^{s} \alpha_{ikl}\omega_{ikl}$$

Proof.

$$\begin{aligned}
\frac{d}{d\lambda} L_M(\lambda)\big|_{\lambda=0} &= \frac{1}{s} \sum_{k=1}^{s} \frac{d}{d\lambda} L(M(\mathbf{x}_0^k, \lambda), \mathbf{y}^k)\big|_{\lambda=0} \\
&= \frac{1}{s} \sum_{k=1}^{s} \frac{d}{d\mathbf{x}} L(\mathbf{x}, \mathbf{y}^k)^T\big|_{\mathbf{x}=M(\mathbf{x}_0^k,0)} \frac{d}{d\lambda} M(\mathbf{x}_0^k, \lambda)\big|_{\lambda=0} \\
&= \frac{1}{s} \sum_{k=1}^{s} \frac{d}{d\mathbf{x}} L(\mathbf{x}, \mathbf{y}^k)^T\big|_{\mathbf{x}=\mathbf{x}_n^k} \frac{d}{d\lambda} M(\mathbf{x}_0^k, \lambda)\big|_{\lambda=0} \qquad \text{which we denote by} \\
&= \frac{1}{s} \sum_{k=1}^{s} \frac{d}{d\mathbf{x}_n^k} L(\mathbf{x}_n^k)^T \frac{d}{d\lambda} M(\mathbf{x}_0^k, \lambda)\big|_{\lambda=0}
\end{aligned}$$

The second line comes from applying the chain rule. We continue by focusing on the summand. The first term, $\frac{d}{d\mathbf{x}_n^k} L(\mathbf{x}_n^k)^T$, does not simplify. For the second term, note that $M(\mathbf{x}, \lambda) = \prod_{i=1}^{n} (W_i - \lambda \frac{\partial L_M}{\partial W_i})\mathbf{x}$. Therefore by the generalized product rule,

$$\frac{d}{dx} \prod_{j=1}^{n} f_j(x) = \sum_{i=1}^{n} \prod_{j=i+1}^{n} \big(f_j(x)\big) f_i'(x) \prod_{k=1}^{i-1} \big(f_k(x)\big)$$

we have

$$\begin{aligned}
&\frac{d}{d\lambda} M(\mathbf{x}_0^k, \lambda)\big|_{\lambda=0} \\
&= -\sum_{i=1}^{n} \prod_{j=i+1}^{n} \big(W_j - \lambda \tfrac{\partial L_M}{\partial W_j}\big) \tfrac{\partial L_M}{\partial W_i} \prod_{l=1}^{i-1} \big(W_l - \lambda \tfrac{\partial L_M}{\partial W_l}\big)\mathbf{x}_0^k\big|_{\lambda=0} \\
&= -\sum_{i=1}^{n} W_{n,i+1}\big(\tfrac{\partial L_M}{\partial W_i}\big) W_{i-1,1}\mathbf{x}_0^k \\
&= -\sum_{i=1}^{n} W_{n,i+1}\big(\tfrac{\partial L_M}{\partial P_i} Hv(\mathbf{x}_{i-1}^k) + \tfrac{\partial L_M}{\partial N_i} Hv(-\mathbf{x}_{i-1}^k)\big) W_{i-1,1}\mathbf{x}_0^k \\
&= -\sum_{i=1}^{n} W_{n,i+1}\Big(\tfrac{1}{s}\sum_{l_1=1}^{s} \tfrac{\partial}{\partial P_i} L(\mathbf{x}_n^{l_1}) Hv(\mathbf{x}_{i-1}^k) + \tfrac{1}{s}\sum_{l_2=1}^{s} \tfrac{\partial}{\partial N_i} L(\mathbf{x}_n^{l_2}) Hv(-\mathbf{x}_{i-1}^k)\Big)\mathbf{x}_{i-1}^k \\
&= -\sum_{i=1}^{n} W_{n,i+1}\Big(\tfrac{1}{s}\sum_{l=1}^{s} \tfrac{\partial}{\partial P_i} L(\mathbf{x}_n^{l}) Hv(\mathbf{x}_{i-1}^k) + \tfrac{\partial}{\partial N_i} L(\mathbf{x}_n^{l}) Hv(-\mathbf{x}_{i-1}^k)\Big)\mathbf{x}_{i-1}^k \\
&= -\tfrac{1}{s}\sum_{i=1}^{n} W_{n,i+1}\sum_{l=1}^{s} \tfrac{\partial}{\partial \mathbf{x}_{i+1}^l} L(\mathbf{x}_n^l)(\mathbf{x}_{i-1}^l)^T \Big(Hv(\mathbf{x}_{i-1}^l) Hv(\mathbf{x}_{i-1}^k) + Hv(-\mathbf{x}_{i-1}^l) Hv(-\mathbf{x}_{i-1}^k)\Big)\mathbf{x}_{i-1}^k \\
&= -\tfrac{1}{s}\sum_{i=1}^{n} W_{n,i+1}\sum_{l=1}^{s} \tfrac{\partial}{\partial \mathbf{x}_{i+1}^l} L(\mathbf{x}_n^l) \sum_{m=1}^{d_i} \mathrm{ReLU}((\mathbf{x}_{i-1}^l)^T \mathbf{e}_m \mathbf{e}_m^T \mathbf{x}_{i-1}^k)
\end{aligned}$$

The last step follows from Lemma 4.4. Combining everything gives

$$
\begin{aligned}
&\frac{d}{d\lambda} L_M(\lambda)\big|_{\lambda=0} \\
&= -\tfrac{1}{s^2} \sum_{k=1}^{s} \tfrac{\partial}{\partial \mathbf{x}_n^k} L(\mathbf{x}_n^k) \sum_{i=1}^{n} W_{n,i+1} \sum_{l=1}^{s} \tfrac{\partial}{\partial \mathbf{x}_{i+1}^l} L(\mathbf{x}_n^l) \sum_{m=1}^{d_n} \mathrm{ReLU}((\mathbf{x}_{i-1}^l)^T \mathbf{e}_m \mathbf{e}_m^T \mathbf{x}_{i-1}^k) \\
&= -\tfrac{1}{s^2} \sum_{k=1}^{s} \sum_{i=1}^{n} \sum_{l=1}^{s} \tfrac{\partial}{\partial \mathbf{x}_n^k} L(\mathbf{x}_n^k)^T W_{n,i+1} (W_{n,i+1})^T \tfrac{\partial}{\partial \mathbf{x}_n^l} L(\mathbf{x}_n^l)^T \sum_{m=1}^{d_i} \mathrm{ReLU}((\mathbf{x}_{i-1}^l)^T \mathbf{e}_m \mathbf{e}_m^T \mathbf{x}_{i-1}^k) \\
&= -\tfrac{1}{s^2} \sum_{k=1}^{s} \sum_{i=1}^{n} \sum_{l=1}^{s} \omega_{ikl} \sum_{m=1}^{d_i} \mathrm{ReLU}\big((\mathbf{x}_0^l)^T W_{i-1,1}^T \mathbf{e}_m \mathbf{e}_m^T W_{i-1,1} \mathbf{x}_0^k\big) \\
&= -\tfrac{1}{s^2} \sum_{k=1}^{s} \sum_{i=1}^{n} \sum_{l=1}^{s} \alpha_{ikl} \omega_{ikl}
\end{aligned}
$$

□

Later in Theorem 4.13, we will apply the expectation to this summation, and also split $\mathrm{E}[\alpha_{ikl}\omega_{ikl}]$ into $\mathrm{E}[\alpha_{ikl}]$ and $\mathrm{E}[\omega_{ikl}]$. The next few lemmata will be used to evaluate $\mathrm{E}[\omega_{ikl}]$.

Lemma 4.7. *(Hettinger) Let $A \in \mathbb{R}^{p\times q}$ be a random matrix such that $a_{ij} \sim \mathcal{N}(0,1)$. Then*

$$\mathrm{E}\big[A^T A\big] = pI$$

Proof. Let A_i be the ith column of A. Then the diagonal elements of $A^T A$ are $A_i^T A_i$ and the non-diagonal elements are $A_i^T A_j$ for $i \neq j$. Since $A_i^T A_i$ is the sum of the squares of p independent standard normal variables, we have $\mathrm{E}\big[A_i^T A_i\big] = p$. For $A_i^T A_j = a_{1i}a_{1j} + a_{2i}a_{2j} + \cdots + a_{pi}a_{pj}$, each term is the product of independent normal values, so $\mathrm{E}[a_{ki}a_{kj}] = \mathrm{E}[a_{ki}]\,\mathrm{E}[a_{kj}] = 0$.

Lemma 4.8. *Let $A \in \mathbb{R}^{p\times q}$ be a random matrix such that $a_{ij} \sim \mathcal{N}(0,1)$. Let $\mathbf{x}, \mathbf{y} \in \mathbb{R}^q$. Then*

$$\mathrm{E}\big[\mathbf{x}^T A^T A \mathbf{y}\big] = p\mathbf{x}^T\mathbf{y}$$

Proof.

$$\begin{aligned}
\mathrm{E}\big[\mathbf{x}^T A^T A\mathbf{y}\big] &= \mathbf{x}^T\,\mathrm{E}\big[A^T A\big]\mathbf{y} \\
&= \mathbf{x}^T(pI)\mathbf{y} && \text{Lemma 4.7} \\
&= p\mathbf{x}^T\mathbf{y} && \square
\end{aligned}$$

The following lemma is the evaluation of $\mathrm{E}[\omega_{ikl}]$ if all layers are initialized as the standard normal. The scaled result is shown in Lemma 4.10.

Lemma 4.9. *Let $m_{q-1}, m_q, \ldots, m_s \in \mathbb{N}$. For $q \leq r \leq s$, let A_r be an $m_r \times m_{r-1}$ matrix. Let all of the elements of A_r be independent and identically distributed as the standard normal distribution. Let $B_{k,l} = A_k A_{k-1} \cdots A_l$ for $k \geq l$. Let $\mathbf{x}, \mathbf{y} \in \mathbb{R}^{m_0}$. Then*

$$\mathrm{E}\big[\mathbf{x}^T B_{s,q}(B_{s,q})^T\mathbf{y}\big] = m_{q-1}m_q \cdots m_{s-1}\mathbf{x}^T\mathbf{y}$$

Proof. We use induction. For convenience, the base case is $q = s$, and the inductive step is performed by decrementing q. The base case is when there is a single A_s^T matrix. Since A_s^T has dimensions $m_{s-1} \times m_s$, we have by Lemma 4.8 that $\mathrm{E}\big[\mathbf{x}^T A_s A_s^T\mathbf{y}\big] = m_{s-1}\mathbf{x}^T\mathbf{y}$ as desired.

For the inductive hypothesis, assume

$$\mathrm{E}\big[\mathbf{x}^T B_{s,q+1}(B_{s,q+1})^T\mathbf{y}\big] = m_q \cdots m_{s-1}\mathbf{x}^T\mathbf{y}$$

Apply Adam's Law to separate A_q, and let $\mathbf{v} = (B_{s,q+1})^T\mathbf{x}$ and $\mathbf{w} = (B_{s,q+1})^T\mathbf{y}$. Then

$$
\begin{aligned}
\mathrm{E}\big[\mathbf{x}^T B_{s,q}(B_{s,q})^T\mathbf{y}\big] &= \mathrm{E}\big[\mathrm{E}\big[\mathbf{x}^T B_{s,q+1}A_q A_q^T(B_{s,q+1})^T\mathbf{y}|B_{s,q+1}\big]\big] \\
&= \mathrm{E}\big[\mathrm{E}\big[\mathbf{v}^T A_q A_q^T\mathbf{w}|B_{s,q+1}\big]\big] && \text{Lemma 4.8} \\
&= \mathrm{E}\big[m_{q-1}\mathbf{v}^T\mathbf{w}\big] \\
&= m_{q-1}\,\mathrm{E}\big[\mathbf{x}^T B_{s,q+1}(B_{s,q+1})^T\mathbf{y}\big] && \text{Inductive Hypothesis} \\
&= m_{q-1}m_q\cdots m_{s-1}\mathbf{x}^T\mathbf{y} && \square
\end{aligned}
$$

Lemma 4.10. *Consider the statement of Lemma 4.9, except that each element of A_i is distributed as $\mathcal{N}(0,\sigma_i^2)$. We have*

$$\mathrm{E}\big[\mathbf{x}^T B_{s,q}(B_{s,q})^T\mathbf{y}\big] = \sigma_q^2\cdots\sigma_s^2 m_{q-1}\cdots m_{s-1}\mathbf{x}^T\mathbf{y},$$

and when $\sigma_i = (m_i m_{i-1})^{-1/4}$, we have

$$\mathrm{E}\big[\mathbf{x}^T B_{s,q}(B_{s,q})^T\mathbf{y}\big] = \frac{\sqrt{m_{q-1}}}{\sqrt{m_s}}\mathbf{x}^T\mathbf{y}.$$

Proof. Note that if $x\sim\mathcal{N}(0,1)$, then $\sigma x\sim\mathcal{N}(0,\sigma^2)$. Thus we need only scale the result from Lemma 4.9 by the appropriate factor. Each matrix $A_q,\ldots,A_s$ appears twice in the expression, thus the factor is $\sigma_q^2\cdots\sigma_s^2$, which immediately gives us the first result. Now we make the substitution $\sigma_i = (m_i m_{i-1})^{-1/4}$.

$$
\begin{aligned}
\mathrm{E}\big[\mathrm{ReLU}(\mathbf{x}^T B_{s,q}(B_{s,q})^T\mathbf{y})\big] &= \sigma_q^2\cdots\sigma_s^2 m_{q-1}\cdots m_{s-1}\mathbf{x}^T\mathbf{y} \\
&= \frac{m_{q-1}\cdots m_{s-1}}{\sqrt{m_{q-1}}m_q\cdots m_{s-1}\sqrt{m_s}}\mathbf{x}^T\mathbf{y} \\
&= \frac{\sqrt{m_{q-1}}}{\sqrt{m_s}}\mathbf{x}^T\mathbf{y} && \square
\end{aligned}
$$

The following lemma is evaluating $\mathrm{E}[\alpha_{ikl}]$. In my opinion, this following calculation is

the most interesting one of this paper. It was complicated by the ReLU function, but we can still compute the expectation exactly.

Lemma 4.11. *Let $m_0, \dots, m_s \in \mathbb{N}$. For $1 \leq q \leq s$, let A_q be an $m_q \times m_{q-1}$ matrix. Let all of the elements of A_q be independent and identically distributed as the standard normal distribution. Denote by a_{qij} the element in the ith row and the jth column of A_q. Let $B_{q,r} = A_q \cdots A_r$ for $q \geq r$. Let $1 \leq k \leq m_s$. Let $\mathbf{x}, \mathbf{y} \in \mathbb{R}^{m_0}$. Let θ be the angle between $\mathbf{x}$ and $\mathbf{y}$ such that $0 \leq \theta \leq \pi$. Then*

$$\sum_{k=1}^{m_s} \mathrm{E}\big[\mathrm{ReLU}(\mathbf{x}^T (B_{s,1})^T \mathbf{e}_k \mathbf{e}_k^T (B_{s,1}) \mathbf{y})\big] = m_1 m_2 \cdots m_s \frac{\|\mathbf{x}\| \, \|\mathbf{y}\|}{\pi} \big((\pi - \theta)\cos\theta + \sin\theta\big).$$

Proof. We first work on the summand using Adam's Law to split this expectation between A_1 and $B_{s,2}$:

$$\mathrm{E}\big[\mathrm{ReLU}(\mathbf{x}^T (B_{s,1})^T \mathbf{e}_k \mathbf{e}_k^T (B_{s,1}) \mathbf{y})\big] = \mathrm{E}\big[\mathrm{E}\big[\mathrm{ReLU}(\mathbf{x}^T (B_{s,1})^T \mathbf{e}_k \mathbf{e}_k^T (B_{s,1}) \mathbf{y}) | B_{s,2}\big]\big]. \tag{4.6}$$

We focus on the inner expectation first. Let $\mathbf{z} = (B_{s,2})^T \mathbf{e}_k$.

$$\begin{aligned}
\mathrm{E}\big[\mathrm{ReLU}(\mathbf{x}^T (B_{s,1})^T \mathbf{e}_k \mathbf{e}_k^T (B_{s,1}) \mathbf{y}) | B_{s,2}\big] &= \mathrm{E}\big[\mathrm{ReLU}(\mathbf{x}^T A_1^T (B_{s,2})^T \mathbf{e}_k \mathbf{e}_k^T (B_{s,2}) A_1 \mathbf{y}) | B_{s,2}\big] \\
&= \mathrm{E}\big[\mathrm{ReLU}(\mathbf{x}^T A_1^T \mathbf{z}\mathbf{z}^T A_1 \mathbf{y}) | B_{s,2}\big]
\end{aligned}$$

By Lemma 4.2, we have that for orthonormal P and Q, PA_1Q has the same distribution as A_1. Thus, we can replace A_1 with PA_1Q in the expectation for any particular choice of P and Q. Choose P such that $P^T \mathbf{z} = \|\mathbf{z}\| \, \mathbf{e}_1$. And choose Q such that $Q\mathbf{x} = \|\mathbf{x}\| \, \mathbf{e}_1$ and $Q\mathbf{y} = \|\mathbf{y}\| \, \mathbf{w}$ where $\mathbf{w} = [\cos\theta, \sin\theta, 0, \dots, 0]^T$. These P and Q exist and can be constructed with the Full QR decomposition algorithm. Specifically, P is the resulting orthonormal

matrix from using Full QR on just the vector $\mathbf{z}$, and Q is the transpose of the resulting orthonormal matrix from using Full QR on $[\mathbf{x}|\mathbf{y}]$.

Also note that $\text{ReLU}(ax) = a \cdot \text{ReLU}(x)$ for $a \geq 0$.

$$
\begin{aligned}
&\mathrm{E}\big[\text{ReLU}(\mathbf{x}^T A_1^T \mathbf{z}\mathbf{z}^T A_1 \mathbf{y})|B_{s,2}\big] \\
&= \mathrm{E}\big[\text{ReLU}(\mathbf{x}^T Q^T A_1^T P^T \mathbf{z}\mathbf{z}^T P A_1 Q\mathbf{y})|B_{s,2}\big] \\
&= \|\mathbf{x}\|\,\|\mathbf{y}\|\,\mathrm{E}\big[\|\mathbf{z}\|^2|B_{s,2}\big]\,\mathrm{E}\big[\text{ReLU}(\mathbf{e}_1^T A_1^T \mathbf{e}_1\mathbf{e}_1^T A_1\mathbf{w})|B_{s,2}\big] \\
&= \|\mathbf{x}\|\,\|\mathbf{y}\|\,\mathrm{E}\big[\|\mathbf{z}\|^2|B_{s,2}\big]\,\mathrm{E}[\text{ReLU}(a_{111}(a_{111}\cos\theta + a_{112}\sin\theta))] \\
&= \|\mathbf{x}\|\,\|\mathbf{y}\|\,\mathrm{E}\big[\|\mathbf{z}\|^2|B_{s,2}\big]\,\mathrm{E}\big[\text{ReLU}(a_{111}^2\cos\theta + a_{111}a_{112}\sin\theta)\big] \\
&= \|\mathbf{x}\|\,\|\mathbf{y}\|\,\mathrm{E}\big[\|\mathbf{z}\|^2|B_{s,2}\big]\tfrac{1}{2\pi}\int_{\mathbb{R}^2}\text{ReLU}(x^2\cos\theta + xy\sin\theta)e^{-\frac{x^2+y^2}{2}}dxdy \\
&= \|\mathbf{x}\|\,\|\mathbf{y}\|\,\mathrm{E}\big[\|\mathbf{z}\|^2|B_{s,2}\big]\tfrac{1}{2\pi}\int_0^{2\pi}\int_0^{\infty}\text{ReLU}(r^2\cos^2\phi\cos\theta + r^2\sin\phi\cos\phi\sin\theta)e^{-\frac{r^2}{2}}rdrd\phi \\
&= \|\mathbf{x}\|\,\|\mathbf{y}\|\,\mathrm{E}\big[\|\mathbf{z}\|^2|B_{s,2}\big]\tfrac{1}{2\pi}\Big(\int_0^{\infty}r^3e^{-\frac{r^2}{2}}dr\Big)\Big(\int_0^{2\pi}\text{ReLU}(\cos(2\phi-\theta)+\cos\theta)d\phi\Big) \\
&= \|\mathbf{x}\|\,\|\mathbf{y}\|\,\mathrm{E}\big[\|\mathbf{z}\|^2|B_{s,2}\big]\tfrac{1}{2\pi}(2)\Big(2\int_{\theta-\frac{\pi}{2}}^{\frac{\pi}{2}}\cos(2\phi-\theta)+\cos\theta d\phi\Big) \\
&= \|\mathbf{x}\|\,\|\mathbf{y}\|\,\mathrm{E}\big[\|\mathbf{z}\|^2|B_{s,2}\big]\tfrac{1}{\pi}\big((\pi-\theta)\cos\theta+\sin\theta\big)
\end{aligned}
$$

Now we evaluate the outer expectation from Equation 4.6. Note that $\mathrm{E}\big[\|\mathbf{z}\|^2|B_{s,2}\big]$ is the only term that depends on $B_{s,2}$, so everything else factors out of the outer expectation. Thus, we need only evaluate $\mathrm{E}\big[\|\mathbf{z}\|^2\big]$ since $\mathrm{E}\big[\mathrm{E}\big[\|\mathbf{z}\|^2|B_{s,2}\big]\big] = \mathrm{E}\big[\|\mathbf{z}\|^2\big]$. By Lemma 4.9, we have

$$
\begin{aligned}
\mathrm{E}\big[\|\mathbf{z}\|^2\big] &= \mathrm{E}\big[\mathbf{e}_k^T B_{s,2}(B_{s,2})^T\mathbf{e}_k\big] \\
&= m_1m_2\cdots m_{s-1}\mathbf{e}_k^T\mathbf{e}_k \\
&= m_1m_2\cdots m_{s-1}
\end{aligned}
$$

Thus we have the result

$$\sum_{k=1}^{m_s} \mathrm{E}\big[\mathrm{ReLU}(\mathbf{x}^T(B_{s,1})^T\mathbf{e}_k\mathbf{e}_k^T(B_{s,1})\mathbf{y})\big] = \sum_{k=1}^{m_s} m_1m_2\cdots m_{s-1}\frac{\|\mathbf{x}\|\,\|\mathbf{y}\|}{\pi}\big((\pi-\theta)\cos\theta+\sin\theta\big)$$
$$= m_1m_2\cdots m_s\frac{\|\mathbf{x}\|\,\|\mathbf{y}\|}{\pi}\big((\pi-\theta)\cos\theta+\sin\theta\big) \qquad \square$$

Corollary 4.12. *Consider the statement of Lemma 4.13, except each element of A_q is distributed as $\mathcal{N}(0,\sigma_q^2)$, then*

$$\mathrm{E}\big[\mathrm{ReLU}(\mathbf{x}^T(B_{s,1})^T\mathbf{e}_k\mathbf{e}_k^T(B_{s,1})\mathbf{y})\big] = \sigma_1^2\sigma_2^2\ldots\sigma_n^2m_1m_2\cdots m_s\frac{\|\mathbf{x}\|\,\|\mathbf{y}\|}{\pi}\big((\pi-\theta)\cos\theta+\sin\theta\big),$$

and when $\sigma_q = (m_qm_{q-1})^{-1/4}$, we have

$$\mathrm{E}\big[\mathrm{ReLU}(\mathbf{x}^T(B_{s,1})^T\mathbf{e}_k\mathbf{e}_k^T(B_{s,1})\mathbf{y})\big] = \frac{\sqrt{m_s}}{\sqrt{m_0}}\frac{\|\mathbf{x}\|\,\|\mathbf{y}\|}{\pi}\big((\pi-\theta)\cos\theta+\sin\theta\big).$$

Proof. Note that if $x\sim\mathcal{N}(0,1)$, then $\sigma x\sim\mathcal{N}(0,\sigma^2)$. Thus we need only scale the result from Lemma 4.13 by the appropriate factor. Each matrix $A_1,\ldots,A_s$ appears twice in the expression, thus the factor is $\sigma_1^2\sigma_2^2\ldots\sigma_s^2$, which immediately gives us the first result. Now we make the substitution.

$$\mathrm{E}\big[\mathrm{ReLU}(\mathbf{x}^T(B_{s,1})^T\mathbf{e}_k\mathbf{e}_k^T(B_{s,1})\mathbf{y})\big] = \sigma_1^2\sigma_2^2\ldots\sigma_s^2m_1m_2\cdots m_s\frac{\|\mathbf{x}\|\,\|\mathbf{y}\|}{\pi}\big((\pi-\theta)\cos\theta+\sin\theta\big)$$
$$= \frac{m_1m_2\cdots m_s}{\sqrt{m_0}m_1\ldots m_{s-1}\sqrt{m_s}}\frac{\|\mathbf{x}\|\,\|\mathbf{y}\|}{\pi}\big((\pi-\theta)\cos\theta+\sin\theta\big)$$
$$= \frac{\sqrt{m_s}}{\sqrt{m_0}}\frac{\|\mathbf{x}\|\,\|\mathbf{y}\|}{\pi}\big((\pi-\theta)\cos\theta+\sin\theta\big) \qquad \square$$

Theorem 4.13. *Let θ_{kl} be the angle between $\mathbf{x}_0^l$ and $\mathbf{x}_0^k$. Then*

$$\mathrm{E}\left[\frac{d}{d\lambda}L_M(0)\right] \approx -\tfrac{1}{s^2}\sum_{i=1}^{n}\frac{\sqrt{d_i d_{i-1}}}{\sqrt{d_0 d_n}}\sum_{k=1}^{s}\sum_{l=1}^{s}\tfrac{\partial}{\partial \mathbf{x}_n^l}L(\mathbf{x}_n^l)^T\tfrac{\partial}{\partial \mathbf{x}_n^k}L(\mathbf{x}_n^k)\tfrac{\|\mathbf{x}_0^k\|\|\mathbf{x}_0^l\|}{\pi}((\pi-\theta_{kl})\cos\theta_{kl}+\sin\theta_{kl})$$

Proof. From Lemma 4.6, we have

$$\mathrm{E}\left[\frac{d}{d\lambda}L_M(0)\right] = -\tfrac{1}{s^2}\sum_{i=1}^{n}\sum_{k=1}^{s}\sum_{l=1}^{s}\mathrm{E}[\alpha_{ikl}\omega_{ikl}]$$

Now we need only evaluate the summand. Here we make the only assumption in the calculation of the first derivative expectation. We treat $\frac{\partial}{\partial \mathbf{x}_n^l}L(\mathbf{x}_n^l)$ and $\frac{\partial}{\partial \mathbf{x}_n^k}L(\mathbf{x}_n^k)$ as random variables. This is only reasonable for a network early in its training. That is when we expect the network to produce meaningless output. We have that α_{ikl} depends only on $W_{i-1,1}$, and under this assumption we have that ω_{ikl} depends only on $W_{n,i+1}$. Thus α_{ikl} is approximately independent of ω_{ikl}

$$\mathrm{E}[\alpha_{ikl}\omega_{ikl}] \approx \mathrm{E}[\alpha_{ikl}]\,\mathrm{E}[\omega_{ikl}]$$

By Lemma 4.12, we have

$$\mathrm{E}[\alpha_{ikl}] = \frac{\sqrt{d_{i-1}}}{\sqrt{d_0}}\frac{\|\mathbf{x}_0^k\|\|\mathbf{x}_0^l\|}{\pi}((\pi-\theta_{kl})\cos\theta_{kl}+\sin\theta_{kl}).$$

By Lemma 4.10, we have

$$\mathrm{E}[\omega_{ikl}] = \frac{\sqrt{d_i}}{\sqrt{d_n}}\tfrac{\partial}{\partial \mathbf{x}_n^l}L(\mathbf{x}_n^l)^T\tfrac{\partial}{\partial \mathbf{x}_n^k}L(\mathbf{x}_n^k).$$

Combining these with the summation gives

$$\mathrm{E}\left[\frac{d}{d\lambda}L_M(0)\right] \approx -\tfrac{1}{s^2}\sum_{i=1}^{n}\frac{\sqrt{d_i d_{i-1}}}{\sqrt{d_0 d_n}}\sum_{k=1}^{s}\sum_{l=1}^{s}\tfrac{\partial}{\partial \mathbf{x}_n^l}L(\mathbf{x}_n^l)^T\tfrac{\partial}{\partial \mathbf{x}_n^k}L(\mathbf{x}_n^k)\tfrac{\|\mathbf{x}_0^k\|\|\mathbf{x}_0^l\|}{\pi}((\pi-\theta_{kl})\cos\theta_{kl}+\sin\theta_{kl}).$$

4.6 Second Derivative Expectation

Now we get the expression for the second derivative.

Definition 4.14. Let $H(\mathbf{x})$ be the hessian of the loss function with respect to $\mathbf{x}$, so

$$H(\mathbf{x}) = \tfrac{d^2}{d\mathbf{x}^2} L(\mathbf{x})$$

We borrow the following convenient expressions used by Hettinger.

Definition 4.15. (Hettinger) Define the following scalar values

$$\gamma_{ijklm} = (\mathbf{x}_0^l)^T W_{i-1,1}^T \Big(Hv(\mathbf{x}_{i-1}^l) Hv(\mathbf{x}_{i-1}^k) + Hv(-\mathbf{x}_{i-1}^l) Hv(-\mathbf{x}_{i-1}^k) \Big) W_{i-1,j+1} (W_{n,j+1})^T \tfrac{d}{d\mathbf{x}_n^m} L(x_n^m)$$

$$\chi_{ijklm} = \tfrac{d}{d\mathbf{x}_n^m} L(x_n^m)^T W_{n,j+1} W_{n,j+1}^T H(\mathbf{x}_n^k) W_{n,i+1} W_{n,i+1}^T \tfrac{d}{d\mathbf{x}_n^l} L(x_n^l)$$

These scalar values are chosen because they form the most natural way to break down the later computations.

Lemma 4.16. *(Hettinger)*

$$\begin{aligned} \frac{d^2}{d\lambda^2} L_M(0) =& \frac{2}{s^3} \sum_{i=1}^{n} \sum_{j=1}^{i-1} \sum_{k=1}^{s} \sum_{l=1}^{s} \sum_{m=1}^{s} \alpha_{jkl} \gamma_{ijklm} \omega_{ikl} \\ &+ \frac{1}{s^3} \sum_{i=1}^{n} \sum_{j=1}^{n} \sum_{k=1}^{s} \sum_{l=1}^{s} \sum_{m=1}^{s} \alpha_{jkm} \chi_{ijklm} \alpha_{ikl} \end{aligned}$$

Proof. See Hettinger's Lemma 4.16 [1]. In his proof, Hettinger starts with

$$\frac{d^2}{d\lambda^2} L_M(0) = \frac{1}{s} \sum_{k=1}^{s} \tfrac{d}{d\mathbf{x}_n^k} L(\mathbf{x}_n^k)^T \big(\tfrac{d^2}{d\lambda^2} \mathbf{x}_n^k(0) \big) + \frac{1}{s} \sum_{k=1}^{s} \big(\frac{d}{d\lambda} \mathbf{x}_n^k(0) \big)^T H(\mathbf{x}_n^k) \big(\frac{d}{d\lambda} \mathbf{x}_n^k(0) \big)$$

and proceeds to show

$$\frac{1}{s}\sum_{k=1}^{s}\tfrac{d}{d\mathbf{x}_n^k}L(\mathbf{x}_n^k)^T\big(\tfrac{d^2}{d\lambda^2}\mathbf{x}_n^k(0)\big)=\frac{2}{s^3}\sum_{i=1}^{n}\sum_{j=1}^{i-1}\sum_{k=1}^{s}\sum_{l=1}^{s}\sum_{m=1}^{s}\alpha_{jkl}\gamma_{ijkl}\omega_{ikl}$$

and

$$\frac{1}{s}\sum_{k=1}^{s}\big(\frac{d}{d\lambda}\mathbf{x}_n^k(0)\big)^T H(\mathbf{x}_n^k)\big(\frac{d}{d\lambda}\mathbf{x}_n^k(0)\big)=\frac{1}{s^3}\sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{s}\sum_{l=1}^{s}\sum_{m=1}^{s}\alpha_{jkm}\chi_{ijklm}\alpha_{ikl}.\quad\square$$

Lemma 4.17. *(Hettinger) When a* CReLU *network is initialized with the Hettinger initialization, the expected value of the first sum from Lemma 4.16 is zero.*

$$\mathrm{E}\left[\tfrac{1}{s}\sum_{k=1}^{s}\tfrac{d}{d\mathbf{x}_n^k}L(\mathbf{x}_n^k)^T\big(\tfrac{d^2}{d\lambda^2}\mathbf{x}_n^k(0)\big)\right]=0$$

Proof. See Hettinger's Lemma 4.17 [1]. In his proof, Hettinger shows that $\mathrm{E}[\alpha_{jkl}\gamma_{ijkl}\omega_{ikl}]=0$, hence the entire first summation goes to zero. □

Now we prepare to evaluate the expectation $\mathrm{E}[\alpha_{jkm}\chi_{ijklm}\alpha_{ikl}]$, which is how we evaluate the expectation of the second summation. The challenge in this evaluation comes from the two α terms, which both contain a ReLU function. The full computation is broken down in the following manner. To compute $\mathrm{E}[\alpha_{jkm}\chi_{ijklm}\alpha_{ikl}]$, we use Adam's Law to get $\mathrm{E}[\alpha_{jkm}\chi_{ijklm}\alpha_{ikl}]=\mathrm{E}[\chi_{ijklm}\,\mathrm{E}[\alpha_{jkm}\alpha_{ikl}|\chi_{ijklm}]]$. We work on $\mathrm{E}[\alpha_{jkm}\alpha_{ikl}|\chi_{ijklm}]$ by taking the expectation with respect to one weight matrix at a time. We start with A_{i-1} as the base case, which is solved in Lemma 4.19. The expectation with respect to each subsequent matrix from A_{i-2} to A_1 is evaluated by applying Lemma 4.20 once per matrix. These two lemmata are combined in Lemma 4.21, where we also use Hettinger's work to finish the evaluation of the outer expectation. The final result is given in Theorem 4.24.

Lemma 4.18. *Let $a, b \sim \mathcal{N}(0,1)$ where a and b are independent. Then*

$$\mathrm{E}[\mathrm{ReLU}(ab)|a] = \tfrac{|a|}{\sqrt{2\pi}}$$

Proof. We prove by cases. If $a \geq 0$, then

$$\begin{aligned}\mathrm{E}[\mathrm{ReLU}(ab)|a] &= a\,\mathrm{E}[\mathrm{ReLU}(b)|a] \\ &= \tfrac{a}{\sqrt{2\pi}} \int_0^\infty b e^{-\frac{b^2}{2}}\, db \\ &= \tfrac{a}{\sqrt{2\pi}} = \tfrac{|a|}{\sqrt{2\pi}}\end{aligned}$$

If $a < 0$, then

$$\begin{aligned}\mathrm{E}[\mathrm{ReLU}(ab)|a] &= -a\,\mathrm{E}[\mathrm{ReLU}(-b)|a] \\ &= \tfrac{-a}{\sqrt{2\pi}} \int_{-\infty}^0 -b e^{-\frac{b^2}{2}}\, db \\ &= \tfrac{-a}{\sqrt{2\pi}} = \tfrac{|a|}{\sqrt{2\pi}}\end{aligned}$$

Thus $\mathrm{E}[\mathrm{ReLU}(ab)|a] = \tfrac{|a|}{\sqrt{2\pi}}$ □

Lemma 4.19. *Let $A \in \mathbb{R}^{q\times s}$. Let all of the elements of A be independent and identically distributed as the standard normal distribution. Let $\mathbf{x}, \mathbf{y}, \mathbf{z} \in \mathbb{R}^s$ such that $\mathbf{x}$, $\mathbf{y}$, and $\mathbf{z}$ are mutually orthogonal. Let $\mathbf{u}, \mathbf{v} \in \mathbb{R}^q$. Let θ be the angle between $\mathbf{u}$ and $\mathbf{v}$.*

$$\mathrm{E}\big[\mathrm{ReLU}(\mathbf{x}^T A^T \mathbf{u}\mathbf{u}^T A\mathbf{y})\mathrm{ReLU}(\mathbf{x}^T A^T \mathbf{v}\mathbf{v}^T A\mathbf{z})\big] = \frac{\|\mathbf{x}\|^2 \|\mathbf{y}\| \|\mathbf{z}\| \|\mathbf{u}\|^2 \|\mathbf{v}\|^2}{2\pi^2}\big((\pi - 2\theta)\cos\theta + 2\sin\theta\big)$$

Proof. By Lemma 4.2, we have that for orthonormal P and Q, PAQ has the same distribution as A. Thus, we can replace A with PAQ in the expectation for any particu-

lar choice of P and Q. Choose P such that $P^T\mathbf{u} = \|\mathbf{u}\|\,\mathbf{e}_1$ and $P^T\mathbf{v} = \|\mathbf{v}\|\,\mathbf{w}$ where $\mathbf{w} = [\cos\theta, \sin\theta, 0, \ldots, 0]^T$. And choose Q such that $Q\mathbf{x} = \|\mathbf{x}\|\,\mathbf{e}_1$, $Q\mathbf{y} = \|\mathbf{y}\|\,\mathbf{e}_2$, and $Q\mathbf{z} = \|\mathbf{z}\|\,\mathbf{e}_3$. These P and Q exist and can be constructed by a Full QR decomposition. Specifically, P is the orthonormal matrix resulting from the Full QR decomposition of the matrix $[\mathbf{u}|\mathbf{v}]$. And Q is the transpose of the orthonormal matrix resulting from a Full QR Decomposition of the matrix $[\mathbf{x}|\mathbf{y}|\mathbf{z}]$. In particular, note that this Q has the specified property because we assume that $\mathbf{x}$, $\mathbf{y}$, and $\mathbf{z}$ are mutually orthogonal.

Also, let $A = [\mathbf{a}_1|\mathbf{a}_2|\cdots|\mathbf{a}_s]$

$$
\begin{aligned}
&\mathrm{E}\big[\mathrm{ReLU}(\mathbf{x}^T A^T \mathbf{u}\mathbf{u}^T A\mathbf{y})\mathrm{ReLU}(\mathbf{x}^T A^T \mathbf{v}\mathbf{v}^T A\mathbf{z})\big] \\
&= \mathrm{E}\big[\mathrm{ReLU}(\mathbf{x}^T Q^T A^T P^T \mathbf{u}\mathbf{u}^T PAQ\mathbf{y})\mathrm{ReLU}(\mathbf{x}^T Q^T A^T P^T \mathbf{v}\mathbf{v}^T PAQ\mathbf{z})\big] \\
&= \|\mathbf{x}\|^2 \|\mathbf{y}\| \|\mathbf{z}\| \|\mathbf{u}\|^2 \|\mathbf{v}\|^2 \mathrm{E}\big[\mathrm{ReLU}(\mathbf{e}_1^T A^T \mathbf{e}_1\mathbf{e}_1^T A\mathbf{e}_2)\mathrm{ReLU}(\mathbf{e}_1^T A^T \mathbf{w}\mathbf{w}^T A\mathbf{e}_3)\big] \\
&= \|\mathbf{x}\|^2 \|\mathbf{y}\| \|\mathbf{z}\| \|\mathbf{u}\|^2 \|\mathbf{v}\|^2 \mathrm{E}\big[\mathrm{ReLU}(a_{11}a_{12})\mathrm{ReLU}(\mathbf{a}_1^T \mathbf{w}\mathbf{w}^T \mathbf{a}_3)\big] \\
&= \|\mathbf{x}\|^2 \|\mathbf{y}\| \|\mathbf{z}\| \|\mathbf{u}\|^2 \|\mathbf{v}\|^2 \mathrm{E}\big[\mathrm{E}[\mathrm{ReLU}(a_{11}a_{12})|\mathbf{a}_1, \mathbf{a}_3]\mathrm{ReLU}(\mathbf{a}_1^T \mathbf{w}\mathbf{w}^T \mathbf{a}_3)\big]
\end{aligned}
$$

In the last step we applied Adam's Law. We now examine the inner expectation $\mathrm{E}[\mathrm{ReLU}(a_{11}a_{12})|\mathbf{a}_1, \mathbf{a}_3]$. Since all elements are mutually independent, we know $\mathrm{E}[\mathrm{ReLU}(a_{11}a_{12})|\mathbf{a}_1, \mathbf{a}_3] = \mathrm{E}[\mathrm{ReLU}(a_{11}a_{12})|a_{11}]$. Thus by Lemma 4.18, $\mathrm{E}[\mathrm{ReLU}(a_{11}a_{12})|\mathbf{a}_1, \mathbf{a}_3] = \frac{|a_{11}|}{\sqrt{2\pi}}$.

Let Q_2 be an orthonormal matrix such that $Q_2^T\mathbf{w} = \mathbf{e}_1$ and $Q_2^T\mathbf{e}_1 = \mathbf{w}$. This Q_2 can be constructed as a Householder Reflector that takes $\mathbf{w}$ to $\mathbf{e}_1$. This works since $\mathbf{w}$ is already of

unit length. Thus by Lemma 4.2, we replace A with $Q_2 A$

$$
\begin{aligned}
&\mathrm{E}\big[\mathrm{E}[\mathrm{ReLU}(a_{11}a_{12})|\mathbf{a}_1,\mathbf{a}_3]\mathrm{ReLU}(\mathbf{a}_1^T\mathbf{w}\mathbf{w}^T\mathbf{a}_3)\big]\\
&=\tfrac{1}{\sqrt{2\pi}}\,\mathrm{E}\big[|a_{11}|\,\mathrm{ReLU}(\mathbf{a}_1^T\mathbf{w}\mathbf{w}^T\mathbf{a}_3)\big]\\
&=\tfrac{1}{\sqrt{2\pi}}\,\mathrm{E}\big[\big|\mathbf{e}_1^TA\mathbf{e}_1\big|\,\mathrm{ReLU}(\mathbf{e}_1^TA^T\mathbf{w}\mathbf{w}^TA\mathbf{e}_3)\big]\\
&=\tfrac{1}{\sqrt{2\pi}}\,\mathrm{E}\big[\big|\mathbf{e}_1^TQ_2A\mathbf{e}_1\big|\,\mathrm{ReLU}(\mathbf{e}_1^TA^TQ_2^T\mathbf{w}\mathbf{w}^TQ_2A\mathbf{e}_3)\big]\\
&=\tfrac{1}{\sqrt{2\pi}}\,\mathrm{E}\big[\big|\mathbf{w}^TA\mathbf{e}_1\big|\,\mathrm{ReLU}(\mathbf{e}_1A^T\mathbf{e}_1\mathbf{e}_1^TA\mathbf{e}_3)\big]\\
&=\tfrac{1}{\sqrt{2\pi}}\,\mathrm{E}[|a_{11}\cos\theta+a_{21}\sin\theta|\,\mathrm{ReLU}(a_{11}a_{13})]\\
&=\tfrac{1}{\sqrt{2\pi}}\,\mathrm{E}[|a_{11}\cos\theta+a_{21}\sin\theta|\,\mathrm{E}[\mathrm{ReLU}(a_{11}a_{13})|a_{11},a_{21}]]\\
&=\tfrac{1}{2\pi}\,\mathrm{E}[|a_{11}\cos\theta+a_{21}\sin\theta|\,|a_{11}|]
\end{aligned}
$$

Now we can evaluate $\mathrm{E}[|a_{11}\cos\theta+a_{21}\sin\theta|\,|a_{11}|]$ using polar coordinates.

$$
\begin{aligned}
\mathrm{E}[|a_{11}\cos\theta+a_{21}\sin\theta|\,|a_{11}|] &= \tfrac{1}{2\pi}\int_0^{2\pi}\int_0^{\infty}|r\cos\phi\cos\theta+r\sin\phi\sin\theta|\,|r\cos\phi|\,e^{-\frac{r^2}{2}}r\,dr\,d\phi\\
&=\tfrac{1}{2\pi}\Big(\int_0^{\infty}r^3e^{-\frac{r^2}{2}}\,dr\Big)\Big(\int_0^{2\pi}|\cos\phi\cos\theta+\sin\phi\sin\theta|\,|\cos\phi|\,d\phi\Big)\\
&=\tfrac{1}{\pi}\Big(\int_0^{2\pi}|\cos(\phi-\theta)|\,|\cos\phi|\,d\phi\Big)\\
&=\tfrac{2}{\pi}\Big(\int_{\theta-\frac{\pi}{2}}^{\frac{\pi}{2}}\cos(\phi-\theta)\cos\phi d\phi-\int_{\frac{\pi}{2}}^{\theta+\frac{\pi}{2}}\cos(\phi-\theta)\cos\phi d\phi\Big)\\
&=\frac{1}{\pi}\big((\pi-2\theta)\cos\theta+2\sin\theta\big)
\end{aligned}
$$

Thus

$$
\mathrm{E}\big[\mathrm{ReLU}(\mathbf{x}^TA^T\mathbf{u}\mathbf{u}^TA\mathbf{y})\mathrm{ReLU}(\mathbf{x}^TA^T\mathbf{v}\mathbf{v}^TA\mathbf{z})\big]=\frac{\|\mathbf{x}\|^2\|\mathbf{y}\|\|\mathbf{z}\|\|\mathbf{u}\|^2\|\mathbf{v}\|^2}{2\pi^2}\big((\pi-2\theta)\cos\theta+2\sin\theta\big)
$$

□

Lemma 4.20. *Let $A \in \mathbb{R}^{q\times s}$ (where $q, s \geq 3$) be a random matrix such that $a_{ij} \sim \mathcal{N}(0,1)$. Let $\mathbf{x}, \mathbf{y}, \mathbf{z} \in \mathbb{R}^s$ such that $\mathbf{x}, \mathbf{y}$, and $\mathbf{z}$ are mutually orthogonal. Then*

$$\mathrm{E}\big[\|A\mathbf{x}\|^2 \|A\mathbf{y}\| \|A\mathbf{z}\|\big] \approx \|\mathbf{x}\|^2 \|\mathbf{y}\| \|\mathbf{z}\| q^2$$

Proof. We start by apply Lemma 4.2 using an orthonormal Q which satisfies $Q\mathbf{x} = \|\mathbf{x}\| \mathbf{e}_1$, $Q\mathbf{y} = \|\mathbf{y}\| \mathbf{e}_2$, and $Q\mathbf{z} = \|\mathbf{z}\| \mathbf{e}_3$. This Q exists because $\mathbf{x}, \mathbf{y}$, and $\mathbf{z}$ are mutually orthogonal, and it can be constructed by applying the Full QR decomposition to the matrix $[\mathbf{x}|\mathbf{y}|\mathbf{z}]$. Thus we can replace A with AQ in the expectation. Also let $A = [\mathbf{a}_1|\cdots|\mathbf{a}_s]$

$$\begin{aligned}
\mathrm{E}\big[\|A\mathbf{x}\|^2 \|A\mathbf{y}\| \|A\mathbf{z}\|\big] &= \mathrm{E}\big[\|AQ\mathbf{x}\|^2 \|AQ\mathbf{y}\| \|AQ\mathbf{z}\|\big] \\
&= \|\mathbf{x}\|^2 \|\mathbf{y}\| \|\mathbf{z}\| \,\mathrm{E}\big[\|A\mathbf{e}_1\|^2 \|A\mathbf{e}_2\| \|A\mathbf{e}_3\|\big] \\
&= \|\mathbf{x}\|^2 \|\mathbf{y}\| \|\mathbf{z}\| \,\mathrm{E}\big[\|\mathbf{a}_1\|^2 \|\mathbf{a}_2\| \|\mathbf{a}_3\|\big] \\
&= \|\mathbf{x}\|^2 \|\mathbf{y}\| \|\mathbf{z}\| \,\mathrm{E}\big[\|\mathbf{a}_1\|^2\big] \,\mathrm{E}[\|\mathbf{a}_2\|] \,\mathrm{E}[\|\mathbf{a}_3\|]
\end{aligned}$$

Since $\mathrm{E}[\|\mathbf{a}_2\|] = \mathrm{E}[\|\mathbf{a}_3\|]$, we need only evaluate $\mathrm{E}\big[\|\mathbf{a}_1\|^2\big]$ and $\mathrm{E}[\|\mathbf{a}_2\|]$.

$$\begin{aligned}
\mathrm{E}\big[\|\mathbf{a}_1\|^2\big] &= \mathrm{E}\big[\mathbf{a}_1^T \mathbf{a}_1\big] \\
&= \mathrm{E}\left[\sum_{i=1}^{q} a_{i1}^2\right] \\
&= \sum_{i=1}^{q} \mathrm{E}\big[a_{i1}^2\big] \\
&= \sum_{i=1}^{q} 1 \\
&= q
\end{aligned}$$

For the second computation, we will use the following notation to deal with the spher-

ical coordinates. Let S_{n-1} be the hypervolume on the edge of the unit hypersphere in n-dimensions. So $S_{n-1} = \frac{n\pi^{\frac{n}{2}}}{\Gamma(\frac{n}{2}+1)} = \frac{2\pi^{\frac{n}{2}}}{\Gamma(\frac{n}{2})}$. And let dS_{n-1} be the relevant jacobian when integrating to get S_{n-1}, hence $dS_{n-1} = \sin^{n-2}\phi_1 \sin^{n-3}\phi_2 \cdots \sin\phi_{n-2} d\phi_1 d\phi_2 \cdots d\phi_{n-1}$.

$$\begin{aligned}
\mathrm{E}[\|\mathbf{a}_2\|] &= \mathrm{E}\left[\sqrt{\mathbf{a}_2^T\mathbf{a}_2}\right] \\
&= (2\pi)^{-\frac{q}{2}} \int_{\mathbb{R}^q} \sqrt{\mathbf{a}_2^T\mathbf{a}_2} e^{-\frac{\mathbf{a}_2^T\mathbf{a}_2}{2}} d\mathbf{a}_2 \\
&= (2\pi)^{-\frac{q}{2}} \int_{\mathbb{R}^q} \sqrt{r^2} e^{-\frac{r^2}{2}} r^{q-1} dr dS_{q-1} \\
&= (2\pi)^{-\frac{q}{2}} \int dS_{q-1} \int_0^\infty r^q e^{-\frac{r^2}{2}} dr \\
&= (2\pi)^{-\frac{q}{2}} S_{q-1} 2^{\frac{q-1}{2}} \Gamma(\tfrac{q+1}{2}) \\
&= \frac{\sqrt{2}\Gamma(\frac{q+1}{2})}{\Gamma(\frac{q}{2})} \\
&\approx \sqrt{q}
\end{aligned}$$

Thus

$$\begin{aligned}
\mathrm{E}\left[\|A\mathbf{x}\|^2 \|A\mathbf{y}\| \|A\mathbf{z}\|\right] &= \|\mathbf{x}\|^2 \|\mathbf{y}\| \|\mathbf{z}\| \mathrm{E}\left[\|\mathbf{a}_1\|^2\right] \mathrm{E}[\|\mathbf{a}_2\|] \mathrm{E}[\|\mathbf{a}_3\|] \\
&\approx \|\mathbf{x}\|^2 \|\mathbf{y}\| \|\mathbf{z}\| (q)(\sqrt{q})(\sqrt{q}) \\
&= \|\mathbf{x}\|^2 \|\mathbf{y}\| \|\mathbf{z}\| q^2
\end{aligned}$$ □

Lemma 4.21. *Let $m_0, \ldots, m_n \in \mathbb{N}$. For $1 \leq q \leq n$, let A_q be an $m_q \times m_{q-1}$ matrix. Let all of the elements of A_q be independent and identically distributed as the standard normal distribution. Denote by a_{qij} the element in the ith row and the jth column of A_q. Let $B_{q,r} = A_q \cdots A_r$ for $q \geq r$. Let $1 \leq i \leq j \leq n$, $1 \leq k \leq m_i$ and $1 \leq l \leq m_j$. Let $\mathbf{x}, \mathbf{y}, \mathbf{z} \in \mathbb{R}^{m_0}$. Let $\mathbf{w}, \mathbf{v} \in \mathbb{R}^{m_n}$ and let $P \in \mathbb{R}^{m_n \times m_n}$ be symmetric. We must assume that $B_{p,1}\mathbf{x}, B_{p,1}\mathbf{y}$, and $B_{p,1}\mathbf{z}$ are large and random enough to be approximated as orthogonal for all*

$1 \leq p \leq i-2$. *Similarly, we assume that* $(B_{j-1,i})^T \mathbf{e}_l$ *and* $\mathbf{e}_k$ *are approximately orthogonal.*

Define

$$
\begin{aligned}
X &= \left(\mathbf{w}^T B_{n,j+1}(B_{n,j+1})^T P B_{n,i+1}(B_{n,i+1})^T \mathbf{v}\right) \\
f_{ij} &= \frac{1}{\pi^2} \|\mathbf{x}\|^2 \|\mathbf{y}\| \|\mathbf{z}\| \prod_{r=1}^{i-1} m_r^2 \\
g_{ij} &= \frac{2\mathbf{w}^T P \mathbf{v} + Tr(P)\mathbf{w}^T \mathbf{v}}{3} \prod_{r=j}^{n-1} m_r(m_r+2) + \frac{\mathbf{w}^T P \mathbf{v} - Tr(P)\mathbf{w}^T \mathbf{v}}{3} \prod_{r=j}^{n-1} m_r(m_r-1) \\
h_{ij} &= \frac{1}{3} \prod_{r=i}^{j-1} m_r(m_r+2) + \frac{2}{3} \prod_{r=i}^{j-1} m_r(m_r-1)
\end{aligned}
$$

Then

$$
\sum_{k=1}^{m_{i-1}} \sum_{l=1}^{m_{j-1}} \mathrm{E}\left[\mathrm{ReLU}\left(\mathbf{x}^T (B_{i-1,1})^T \mathbf{e}_k \mathbf{e}_k^T B_{i-1,1} \mathbf{y}\right) \mathrm{ReLU}\left(\mathbf{x}^T (B_{j-1,1})^T \mathbf{e}_l \mathbf{e}_l^T B_{j-1,1} \mathbf{z}\right) X\right] \approx f_{ij} g_{ij} h_{ij}
$$

Proof. We first work on evaluating the summand by using Adam's Law to split this expectation.

$$
\mathrm{E}\left[\mathrm{ReLU}\left(\mathbf{x}^T (B_{i-1,1})^T \mathbf{e}_k \mathbf{e}_k^T B_{i-1,1} \mathbf{y}\right) \mathrm{ReLU}\left(\mathbf{x}^T (B_{j-1,1})^T \mathbf{e}_l \mathbf{e}_l^T B_{j-1,1} \mathbf{z}\right) X\right] \tag{4.7}
$$

$$
= \mathrm{E}\left[\mathrm{E}\left[\mathrm{ReLU}\left(\mathbf{x}^T (B_{i-1,1})^T \mathbf{e}_k \mathbf{e}_k^T B_{i-1,1} \mathbf{y}\right) \mathrm{ReLU}\left(\mathbf{x}^T (B_{j-1,1})^T \mathbf{e}_l \mathbf{e}_l^T B_{j-1,1} \mathbf{z}\right) | B_{n,i}\right] X\right] \tag{4.8}
$$

We focus on the inner expectation first. To evaluate this expectation, we must first apply Lemma 4.19 with the following assignments:

$$
\begin{aligned}
A &:= A_{i-1} & \mathbf{x} &:= B_{i-2,1} \mathbf{x} \\
\mathbf{u} &:= \mathbf{e}_k & \mathbf{y} &:= B_{i-2,1} \mathbf{y} \\
\mathbf{v} &:= (B_{j-1,i})^T \mathbf{e}_l & \mathbf{z} &:= B_{i-2,1} \mathbf{z}
\end{aligned}
$$

Then by applying Lemma 4.19, we get

$$\mathrm{E}\big[\mathrm{ReLU}\big(\mathbf{x}^T(B_{i-1,1})^T\mathbf{e}_k\mathbf{e}_k^TB_{i-1,1}\mathbf{y}\big)\mathrm{ReLU}\big(\mathbf{x}^T(B_{j-1,1})^T\mathbf{e}_l\mathbf{e}_l^TB_{j-1,1}\mathbf{z}\big)|B_{n,i}\big]$$
$$= \frac{\|B_{i-2,1}\mathbf{x}\|^2\|B_{i-2,1}\mathbf{y}\|\|B_{i-2,1}\mathbf{z}\|\|B_{j-1,i}\mathbf{e}_l\|^2}{2\pi^2}\big((\pi-2\theta)\cos\theta+2\sin\theta\big)$$

where θ is the angle between $(B_{j-1,1})^T\mathbf{e}_l$ and $\mathbf{e}_k$. To proceed, we will assume that these vectors are sufficiently large and therefore will very likely be close to orthogonal (see Lemma 4.3). Hence we will assume $\theta = \frac{\pi}{2}$. So this simplifies to

$$\frac{\|B_{i-2,1}\mathbf{x}\|^2\|B_{i-2,1}\mathbf{y}\|\|B_{i-2,1}\mathbf{z}\|\|B_{j-1,i}\mathbf{e}_l\|^2}{\pi^2}$$

Now we evaluate the outer expectation from (4.8). We perform this expectation in an iterated fashion, which relies on repeated use of Adam's Law, such that we start evaluating with A_{i-2} and work sequentially down to A_1. Here we must again make an assumption regarding orthogonality, specifically we assume that $B_{p,1}\mathbf{x}, B_{p,1}\mathbf{y}$, and $B_{p,1}\mathbf{z}$ are mutually orthogonal for all $1 \leq p \leq i-2$. Each iteration can be computed using Lemma 4.20 since each A_p term is only found in $\|B_{i-2,1}\mathbf{x}\|^2\|B_{i-2,1}\mathbf{y}\|\|B_{i-2,1}\mathbf{z}\|$. This iteration results in a product.

$$
\begin{aligned}
&\sum_{k=1}^{m_{i-1}} \sum_{l=1}^{m_{j-1}} \mathrm{E}\big[\mathrm{ReLU}\big(\mathbf{x}^T (B_{i-1,1})^T \mathbf{e}_k \mathbf{e}_k^T B_{i-1,1} \mathbf{y}\big) \mathrm{ReLU}\big(\mathbf{x}^T (B_{j-1,1})^T \mathbf{e}_l \mathbf{e}_l^T B_{j-1,1} \mathbf{z}\big) X\big] \\
&\approx \tfrac{1}{\pi^2} \sum_{k=1}^{m_{i-1}} \sum_{l=1}^{m_{j-1}} \mathrm{E}\big[\|B_{i-2,1}\mathbf{x}\|^2 \|B_{i-2,1}\mathbf{y}\| \|B_{i-2,1}\mathbf{z}\| \|B_{j-1,i}\mathbf{e}_l\|^2 X\big] \\
&\approx \tfrac{1}{\pi^2} \|\mathbf{x}\|^2 \|\mathbf{y}\| \|\mathbf{z}\| \prod_{p=1}^{i-2} \Big(m_p^2\Big) \sum_{k=1}^{m_{i-1}} \sum_{l=1}^{m_{j-1}} \mathrm{E}\big[\|B_{j-1,i}\mathbf{e}_l\|^2 X\big] \\
&= \tfrac{1}{\pi^2} \|\mathbf{x}\|^2 \|\mathbf{y}\| \|\mathbf{z}\| \prod_{p=1}^{i-2} \Big(m_p^2\Big) m_{i-1} \,\mathrm{E}\left[\sum_{l=1}^{m_{j-1}} \|B_{j-1,i}\mathbf{e}_l\|^2 X\right] \\
&= \tfrac{1}{\pi^2} \|\mathbf{x}\|^2 \|\mathbf{y}\| \|\mathbf{z}\| \prod_{p=1}^{i-2} \Big(m_p^2\Big) m_{i-1} \,\mathrm{E}\big[\|B_{j-1,i}\|_F^2 X\big] \\
&= \tfrac{1}{\pi^2} \|\mathbf{x}\|^2 \|\mathbf{y}\| \|\mathbf{z}\| \prod_{p=1}^{i-1} \Big(m_p^2\Big) \,\mathrm{E}\left[\frac{\|B_{j-1,i}\|_F^2}{m_{i-1}} X\right] \\
&= f_{ij} \,\mathrm{E}\left[\frac{\|B_{j-1,i}\|_F^2}{m_{i-1}} X\right]
\end{aligned}
$$

We can partially evaluate the remaining expectation by using Adam's Law and then applying Hettinger's Lemma A.13 [1].

$$
\begin{aligned}
&\mathrm{E}\left[\frac{\|B_{j-1,i}\|_F^2}{m_{i-1}} X\right] \\
&= \mathrm{E}\left[\frac{\|B_{j-1,i}\|_F^2}{m_{i-1}} \big(\mathbf{w}^T B_{n,j+1} (B_{n,j+1})^T P B_{n,i+1} (B_{n,i+1})^T \mathbf{v}\big)\right] \\
&= \mathrm{E}\left[\frac{\|B_{j-1,i}\|_F^2}{m_{i-1}} \mathrm{E}\big[\mathbf{w}^T B_{n,j+1} (B_{n,j+1})^T P B_{n,i+1} (B_{n,i+1})^T \mathbf{v} | B_{j-1,i}\big]\right] \\
&= \Big(\frac{2\mathbf{w}^T P \mathbf{v} + \mathrm{Tr}(P)\mathbf{w}^T \mathbf{v}}{3} \prod_{r=j}^{n-1} m_r(m_r+2) + \frac{\mathbf{w}^T P \mathbf{v} - \mathrm{Tr}(P)\mathbf{w}^T \mathbf{v}}{3} \prod_{r=j}^{n-1} m_r(m_r-1)\Big) \\
&\mathrm{E}\left[\frac{\|B_{j,i+1}\|_F^2}{m_j} \frac{\|B_{j-1,i}\|_F^2}{m_{i-1}}\right] \\
&= g_{ij} \,\mathrm{E}\left[\frac{\|B_{j,i+1}\|_F^2}{m_j} \frac{\|B_{j-1,i}\|_F^2}{m_{i-1}}\right]
\end{aligned}
$$

At last, we can finish the remaining expectation by applying Lemma A.12 from Hettinger [1], which states that

$$\mathrm{E}\left[\frac{\|B_{j,i+1}\|_F^2}{m_j}\frac{\|B_{j,i-1}\|_F^2}{m_{j-1}}\right] = \frac{1}{3}\prod_{r=i}^{j-1} m_r(m_r+2) + \frac{2}{3}\prod_{r=i}^{j-1} m_r(m_r-1) = h_{ij}.$$

Thus we have the result

$$\sum_{k=1}^{m_{i-1}}\sum_{l=1}^{m_{j-1}} \mathrm{E}\big[\mathrm{ReLU}\big(\mathbf{x}^T(B_{i-1,1})^T\mathbf{e}_k\mathbf{e}_k^T B_{i-1,1}\mathbf{y}\big)\mathrm{ReLU}\big(\mathbf{x}^T(B_{j-1,1})^T\mathbf{e}_l\mathbf{e}_l^T B_{j-1,1}\mathbf{z}\big)X\big] \approx f_{ij}g_{ij}h_{ij} \quad \square$$

Corollary 4.22. *Consider the statement of Lemma 4.21, except $A_{qij} \sim \mathcal{N}(0, \sigma_q^2)$, then*

$$\begin{aligned}&\sum_{k=1}^{m_{i-1}}\sum_{l=1}^{m_{j-1}} \mathrm{E}\big[\mathrm{ReLU}\big(\mathbf{x}^T(B_{i-1,1})^T\mathbf{e}_k\mathbf{e}_k^T B_{i-1,1}\mathbf{y}\big)\mathrm{ReLU}\big(\mathbf{x}^T(B_{j-1,1})^T\mathbf{e}_l\mathbf{e}_l^T B_{j-1,1}\mathbf{z}\big)X\big]\\ &\approx \frac{\sigma_1^4\sigma_2^4\ldots\sigma_n^4}{\sigma_i^2\sigma_j^2} f_{ij}g_{ij}h_{ij}\end{aligned}$$

and when $\sigma_q = (m_q m_{q-1})^{-1/4}$, then

$$\begin{aligned}&\sum_{k=1}^{m_{i-1}}\sum_{l=1}^{m_{j-1}} \mathrm{E}\big[\mathrm{ReLU}\big(\mathbf{x}^T(B_{i-1,1})^T\mathbf{e}_k\mathbf{e}_k^T B_{i-1,1}\mathbf{y}\big)\mathrm{ReLU}\big(\mathbf{x}^T(B_{j-1,1})^T\mathbf{e}_l\mathbf{e}_l^T B_{j-1,1}\mathbf{z}\big)X\big]\\ &\approx \frac{\sqrt{m_i m_{i-1} m_j m_{j-1}}}{m_0 m_1^2\ldots m_{n-1}^2 m_n} f_{ij}g_{ij}h_{ij}\end{aligned}$$

Proof. Note that if $x \sim \mathcal{N}(0,1)$, then $\sigma x \sim \mathcal{N}(0,\sigma^2)$. Thus we need only scale the result from Lemma 4.21 by the appropriate factor. Each matrix $A_1, \ldots, A_n$ appears four times in the expression, with the exception that A_i and A_j only appear twice. Thus the factor is $\frac{\sigma_1^4\sigma_2^4\ldots\sigma_n^4}{\sigma_i^2\sigma_j^2}$, which immediately gives us the first result. Now we make the substitution $\sigma_i = (m_i m_{i-1})^{-1/4}$.

$$\sum_{k=1}^{m_{i-1}}\sum_{l=1}^{m_{j-1}} \mathrm{E}\big[\mathrm{ReLU}\big(\mathbf{x}^T(B_{i-1,1})^T\mathbf{e}_k\mathbf{e}_k^T B_{i-1,1}\mathbf{y}\big)\mathrm{ReLU}\big(\mathbf{x}^T(B_{j-1,1})^T\mathbf{e}_l\mathbf{e}_l^T B_{j-1,1}\mathbf{z}\big)X\big]$$
$$\approx \frac{\sigma_1^4\sigma_2^4\dots\sigma_n^4}{\sigma_i^2\sigma_j^2} f_{ij}g_{ij}h_{ij}$$
$$= \frac{\sqrt{m_i m_{i-1} m_j m_{j-1}}}{m_0 m_1^2\dots m_{n-1}^2 m_n} f_{ij}g_{ij}h_{ij} \qquad \square$$

Definition 4.23. (Modified from Hettinger) Let N be an n-layer CReLU network with input dimension d_0 and layer widths $d_1, d_2, \dots, d_n$. Let $1 \le i, j \le n$.

$$\phi_{cd} = \tfrac{2}{3}\prod_{q=c}^{d-1}\big(1+\tfrac{2}{d_q}\big) + \tfrac{1}{3}\prod_{q=c}^{d-1}\big(1-\tfrac{1}{d_q}\big)$$
$$\psi_{cd} = \tfrac{1}{3}\prod_{q=c}^{d-1}\big(1+\tfrac{2}{d_q}\big) - \tfrac{1}{3}\prod_{q=c}^{d-1}\big(1-\tfrac{1}{d_q}\big)$$

Let $a = \min(i,j)$ and $b = \max(i,j)$. Define

$$A_{klm} = \tfrac{1}{\pi^2}\left\|\mathbf{x}_0^k\right\|^2\left\|\mathbf{x}_0^l\right\|\left\|\mathbf{x}_0^m\right\|$$
$$B_{ij} = \phi_{ab} - \psi_{ab}$$
$$C_{ijklm} = \phi_{bn}\tfrac{d}{d\mathbf{x}_n^l}L(\mathbf{x}_n^l)^T H(x_n^k)\tfrac{d}{d\mathbf{x}_n^k}L(\mathbf{x}_n^k) + \psi_{bn}\mathrm{Tr}(H(\mathbf{x}_n^l))\tfrac{d}{d\mathbf{x}_n^l}L(\mathbf{x}_n^l)^T\tfrac{d}{d\mathbf{x}_n^m}L(\mathbf{x}_n^m)$$

Theorem 4.24. *Assume that $\frac{d}{d\mathbf{x}_n^p}L(\mathbf{x}_n^p)$ and $H(\mathbf{x}_n^p)$ may be treated as independent of the rest of the network. Then in the notation of Definition 4.23, we have*

$$\mathrm{E}\Big[\tfrac{d^2}{d\lambda^2}L_M(0)\Big] \approx \sum_{i=1}^{n}\sum_{j=1}^{n}\frac{\sqrt{d_{i-1}d_i d_{j-1}d_j}}{d_0 d_n}\tfrac{1}{s^3}\sum_{k=1}^{s}\sum_{l=1}^{s}\sum_{m=1}^{s}A_{jkl}B_{ij}C_{ijklm}.$$

Proof. We expressed the second derivative as a summation in Lemma 4.16. The first term of the summation goes to zero as shown in Lemma 4.17. We must evaluate the expectation on the second term, whose summand is $\mathrm{E}[\alpha_{jkm}\chi_{ijklm}\alpha_{jkl}]$. First, we assume that we can

treat $\frac{d}{d\mathbf{x}_n^p} L(\mathbf{x}_n^p)$ and $H(\mathbf{x}_n^p)$ as independent of the rest of the network, like we did in Theorem 4.13. This is reasonable since the network output is meaningless on initialization. Then the result is obtained in Corollary 4.22, though most of the calculations are shown in Lemma 4.19, Lemma 4.20, and Lemma 4.21. Numerous approximations are used to comes to this formula, all of which pertain to the term A_{klm}. □

Chapter 5. Testing

We now have estimates for $\mathrm{E}\left[\frac{d}{d\lambda} L_M(0)\right]$ and $\mathrm{E}\left[\frac{d^2}{d\lambda^2} L_M(0)\right]$. From the earlier quadratic approach with Newton's method, we expect that the optimal learning rate is approximated by

$$\lambda_{optimal} \approx -\frac{\mathrm{E}\left[\frac{d}{d\lambda} L_M(0)\right]}{\mathrm{E}\left[\frac{d^2}{d\lambda^2} L_M(0)\right]}.$$

We now attempt to use this formula in a small scale setting. There are a few practical challenges that we now deal with. Note that the approximation for $\mathrm{E}\left[\frac{d^2}{d\lambda^2} L_M(0)\right]$ has five nested summations, and its term count is $n^2 s^3$ for a network with n layers and s data samples. This is computationally infeasible to evaluate for any interesting problem, and even the toy problem will be troublesome. Therefore, we approximate the summation by sampling uniformly over the indices and adding the samples. In doing so, the only adjustment we need to make is scaling appropriately for the number of samples. This factor is $\frac{n^2 s^3}{\#\text{samples}}$. We can similarly estimate the first derivative by sampling and scaling by $\frac{ns^2}{\#\text{samples}}$.

5.1 First Step Testing

For the dataset, we use the diabetes dataset which is commonly available for machine learning. It consists of only 8 features, shown in Table 5.1.

We choose this dataset partly because the features have no spatial connections, and

Feature	Description
Pregnancies	Number of times pregnant
Blood Pressure	Diastolic blood pressure
Glucose	Plasma glucoes concentration
Skin Thickness	Tricep skin fold thickness
Insulin	2-hour serum insulin
Diabetes Pedigree Function	a score based on family history of diabetes
Age	years
BMI	Body mass Index
Diabetes	Whether the patient has diabetes

Table 5.1: Data Features

therefore a fully connected network is a sensible choice. This is not the ideal dataset to work with because it is low dimensional, but higher dimensional data would prove challenging to run the tests on. Thus, we also choose this dataset for computational reasons. To test the formula for a given network, we apply the sampling technique using 4,000 samples to estimate the first and second derivatives.

In Figure 5.1, we compare the true optimal learning rate against the numerical estimate. These numerical results fit pretty well with our expectations. Notice how smaller networks have a large discrepancy between the true optimal value, and the estimated one. This is expected because the calculation of the second derivative relies heavily on the fact that large random vectors tend to be orthogonal. So the larger the network is, the more accurate the estimate should become. Indeed, the graph shows a converging trend between the estimated value and the true value.

The result is excellent validation for the provided formula, however, it necessarily a practical method. The primary challenge is that we are restricted to fully connected networks. And while there are cases where fully connected networks are a good choice, those cases almost always involve relatively small networks. Large fully connected networks are rather uncommon, and that is the main case where this method of optimizing the learning rate is applicable.

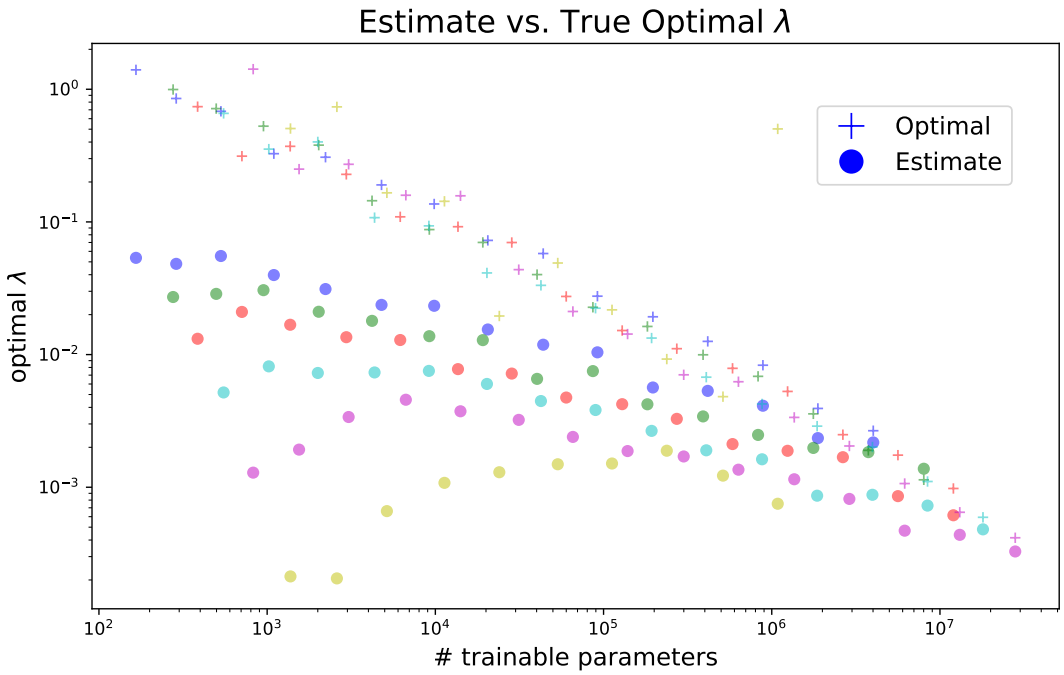

Figure 5.1: Estimated vs True Optimal Learning Rates (Each dot is an estimated optimal learning rate. Each plus is a true optimal learning rate. Each color is a different network depth.)

5.2 More Steps

To this point, we have only considered how the learning rate impacts the first step of optimization. The analysis says nothing about the optimal learning rate after the first step, but we will continue training with the same learning rate for a few more steps and compare results to see how the optimal learning rate for the first step compares. In practice, a learning rate schedule is chosen, which is usually based on the initial learning rate and the amount of improvement in the loss, but we will use a constant learning rate since we are only considering the first 5 steps.

Figure 5.2 shows the results of training for 5 steps on 4 different architectures. Each graph corresponds to a specific network architecture. The green solid line shows the loss when using the optimal learning rate computed above. Blue dotted lines correspond to using smaller learning rates, while red dotted lines are larger than the optimal learning rate.

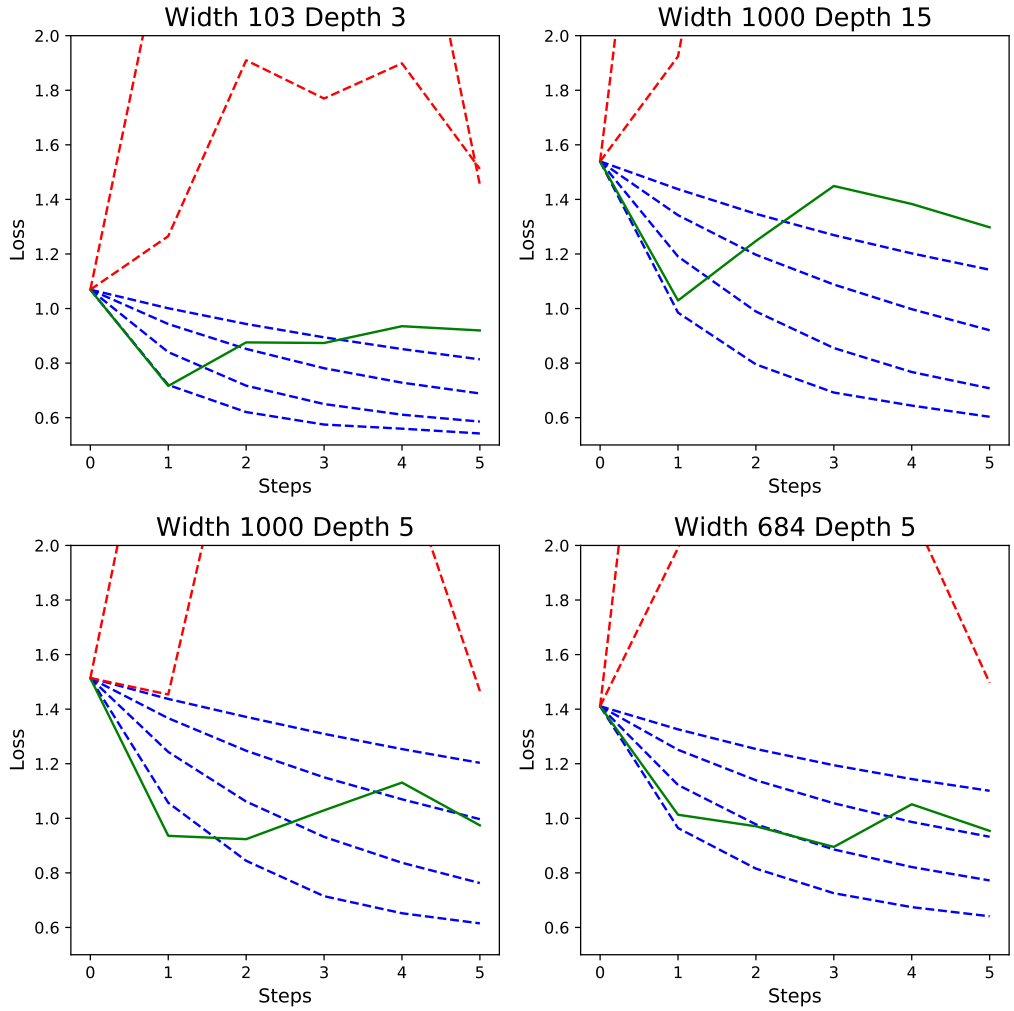

Figure 5.2: Multiple Steps of Training (Each graph is a different architecture. Green line is using the optimal learning rate. Red line is using a larger learning rate. Blue line is using a smaller learning rate.)

All learning rates were chosen by multiplying or dividing the optimal rate by a power of two.

On average, we expect the green line to have the best (lowest) loss at step 1, and this is true for two of the graphs above. The results suggest that the optimal learning rate is somewhat unstable since the slope of the green line always increases after the first step, and increasing the learning rate by at least a factor of two results in a growing loss. Therefore, it seems the better choice is to pick a learning rate slightly smaller than the optimal rate. This should produce more predictable improvement shown in the blue lines. This makes the results seen in Figure 5.1 more interesting since it already suggests that we may want

slightly smaller learning rates, especially in the case of smaller networks.

5.3 Further Research

To make this analysis more practical, it should be extended to other neural network architectures. For example, the convolutional neural network has proven itself as a capable solution to many computer vision problems. Analyzing this network type to estimate its optimal learning rate could yield valuable results.

5.4 Conclusion

Researchers face significant challenges when analyzing and understanding neural networks. My attempts to exactly evaluate the optimal learning proved fruitless, but making the right approximations led to good results in the case we should care most about. Large networks should be the focus when choosing approximations. This area of research is reminiscent of statistical physics where a small set of approximations and tricks are used repeatedly to obtain the results that only apply in the case of large systems. Similarly, we need tricks and approximations that handle large networks.

Bibliography

[1] Christopher James Hettinger. *Hyperparameters for Dense Neural Networks*. PhD thesis, 2019.

[2] Wenling Shang, Kihyuk Sohn, Diogo Almeida, and Honglak Lee. Understanding and improving convolutional neural networks via concatenated rectified linear units. In *international conference on machine learning*, pages 2217–2225, 2016.

[3] Michael Blot, Matthieu Cord, and Nicolas Thome. Max-min convolutional neural networks for image classification. In *2016 IEEE International Conference on Image Processing (ICIP)*, pages 3678–3682. IEEE, 2016.

[4] Jonghong Kim, O Sangjun, Yoonnyun Kim, and Minho Lee. Convolutional neural network with biologically inspired retinal structure. *Procedia Computer Science*, 88:145–154, 2016.

[5] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[6] K. Hornik. Some new results on neural network approximation. *Neural Networks*, 6(8):1069 – 1072, 1993.

[7] David Balduzzi, Marcus Frean, Lennox Leary, JP Lewis, Kurt Wan-Duo Ma, and Brian McWilliams. The shattered gradients problem: If resnets are the answer, then what is the question? In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 342–350. JMLR. org, 2017.

[8] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. *CoRR*, abs/1502.01852, 2015.

[9] Tony Cai, Jianqing Fan, and Tiefeng Jiang. Distributions of angles in random packing on spheres. *The Journal of Machine Learning Research*, 14(1):1837–1864, 2013.