2022-08-04

Gradient Conditioning in Deep Neural Networks

Michael Vernon Nelson
Brigham Young University

Gradient Conditioning in Deep Neural Networks

Michael Vernon Nelson

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
Emily Evans
Jared Whitehead

Department of Mathematics

Brigham Young University

ABSTRACT

Gradient Conditioning in Deep Neural Networks

Michael Vernon Nelson
Department of Mathematics, BYU
Master of Science

When using Stochastic Gradient Descent (SGD) to train Artificial Neural Networks, gradient variance comes from two sources: differences in the weights of the network when each batch gradient is estimated and differences between the input values in each batch. Some architectural traits, like skip-connections and batch-normalization, allow much deeper networks to be trained by reducing each type of variance and improving the conditioning of the network gradient with respect to both the weights and the input. It is still unclear to which degree each property is responsible for these dramatic stability improvements when training deep networks. This thesis summarizes previous findings related to gradient conditioning in each case, demonstrates efficient methods by which each can be measured independently, and investigates the contribution each makes to the stability and speed of SGD in various architectures as network depth increases.

Keywords: gradient conditioning, gradient step-consistency, gradient batch-dissonance, gradient whitening, gradient confusion, gradient coherence, gradient diversity

Acknowledgements

First and foremost I would like to thank Dr. Tyler Jarvis for his continuing support throughout the long process of completing this thesis. I would also like to thank my wife Andi for her incredible patience and support despite all the time that this research has taken me away from home.

Contents

Contents **iv**

List of Figures **vi**

Introduction **1**

1 Gradient Step-Consistency and Batch-Dissonance **2**

1.1 Objectives 5

1.2 Terminology 6

1.3 Notation 7

1.4 What is Conditioning and How is it Related to SGD? 9

1.5 Estimating Gradient Conditioning 13

1.6 Reducing Metric Costs with Random Masking 16

2 Gradient Step-Consistency **16**

2.1 Quantifying Gradient Step-Consistency 17

2.2 Gradient Conditioning with Respect to $\mathbb{W}$ 33

2.3 Generating Targeted Weight Perturbation Samples 37

2.4 Literature Related to Gradient Step-Consistency 42

3 Gradient Batch-Dissonance **49**

3.1 Quantifying Gradient Batch-Dissonance 50

3.2 Gradient Conditioning with Respect to $\mathbb{X}$ 58

3.3 Generating Targeted Input Perturbation Samples 62

3.4 Literature and Metrics on Gradient Batch-Dissonance 66

3.5 Shattered Gradients, the Gradient Rank Metric, and CReLU 68

3.6 Non-Orthogonal Variance and Gradient Confusion 77

3.7 Gradient Diversity 82

3.8 Gradient Coherence 86

4 Remaining Objectives and Future Work **90**

4.1 Adapting Batch-Dissonance Metrics to Measure Gradient Step-Consistency between Steps 91

4.2 Future Work: Architecture and Hyper-Parameter Optimization Tests 100

4.3 Conclusions 105

Appendix A Code **107**

Bibliography **108**

List of Figures

2.1 Simulation Results for Lemma 2.2 31
2.2 Magnitude Bias Simulation Results for Lemma 2.2 32
2.3 Mask Effectiveness for Gradient Conditioning w.r.t. $\mathbb{W}$ 36
2.4 Targeted Sampling methods in $\mathbb{X}$ 40
2.5 Gradient Conditioning w.r.t. the Weight Space Throughout Training 41
2.6 Gradient Conditioning w.r.t. the Weight Space Performance Scatterplot . . . 42

3.1 Mask Effectiveness for Gradient Conditioning w.r.t. $\mathbb{X}$ 61
3.2 Targeted Sampling methods in $\mathbb{X}$ 64
3.3 Gradient Conditioning w.r.t. the Input Space Throughout Training 65
3.4 Gradient Conditioning w.r.t. the Input Space Performance Scatterplot . . . 66
3.5 Balduzzi et al. Gradient Whitening Toy Example 69
3.6 Mask Effectiveness for $\Delta_{\mathbb{X}} R(h_{\mathbb{W}}, w, \chi)$ 71
3.7 Weight Gradient Rank Throughout Training in Various Networks 72
3.8 Weight Gradient Rank Performance Scatterplot 73
3.9 Input Gradient Rank Throughout Training in Various Networks 74
3.10 Input Gradient Rank Performance Scatterplot 75
3.11 Mirrored weights in Looks-Linear Layers Persist 77
3.12 Gradient Confusion Throughout Training in Various Networks 79
3.13 Gradient Confusion Performance Scatterplot 80
3.14 Mask Effectiveness for the Gradient Confusion Metric 82
3.15 Mask Effectiveness for the Gradient Diversity Metric 83
3.16 Gradient Diversity Throughout Training in Various Networks 84
3.17 Gradient Diversity Performance Scatterplot 85
3.18 Mask Effectiveness for Gradient Coherence Metric 88
3.19 Gradient Coherence Throughout Training in Various Networks 89

3.20 Gradient Coherence Performance Scatterplot 90

4.1 Mask Effectiveness for $\Delta_{\mathbb{W}} D_n(w, \chi)$ 93

4.2 Mask Effectiveness for $\Delta_{\mathbb{W}} H_n(w, \chi)$ 93

4.3 Targeted Sampling Methods: Minimum Weight-Gradient Cosine Similarity . 94

4.4 Weight-Gradient Cosine Similarity Throughout Training in Various Networks 95

4.5 Weight-Gradient Cosine Similarity Performance Scatterplot 96

4.6 Weight-Gradient Diversity Throughout Training in Various Networks 97

4.7 Weight-Gradient Diversity Performance Scatterplot 98

4.8 Weight-Gradient Coherence Throughout Training in Various Networks 99

4.9 Weight-Gradient Coherence Performance Scatterplot 100

4.10 Metric Timing by Number of Learnable Parameters 104

Introduction

When using *Stochastic Gradient Descent* (SGD) to train Artificial Neural Networks, variance between batch gradient estimates is unavoidable. Too much variance between batch gradient estimates can slow or prevent convergence. This variance comes from two sources: differences in the weights of the network when each batch gradient is estimated, and differences between the input values within each batch.

Certain architectural traits, such as *skip connections* (subsection 2.4.2), significantly reduce gradient variance between steps during training and enable the training of much deeper networks. These traits have a smoothing effect on the loss surface and reduce variance between gradients estimated at network states that are close in weight space [7, 12, 19]. **In other words, these traits improve the conditioning of the network gradient with respect to weights.** We will use the phrase *gradient step-consistency* or just *step-consistency* when speaking of the degree to which an architecture exhibits this property.

Several papers published since the development of these architectural traits have investigated this ability to smooth the loss surface and increase the stability of SGD in deep networks [12, 19]. However, these papers ignore the potential that architectural traits may have for reducing variance between gradient estimates within each batch and rely solely upon the batch-size parameter to control batch-gradient variance arising from this source.

More recently, a number of papers have demonstrated that high relative variance between individual gradient estimates within a batch results in poor convergence regardless of batch-size [1, 2, 18]. Furthermore, these demonstrate that relative variance between gradient estimates within each batch increases rapidly with network depth for many architectures. Fortunately, skip connections significantly reduce this type of variance and also reduce the rate at which it grows with depth [1, 18]. **In other words, skip connections improve the conditioning of the network gradient with respect to the input.** We will use the phrase *gradient batch-dissonance* or simply *batch-dissonance* when speaking of the degree to which an architecture exhibits this property.

Because skip connections both significantly improves gradient step-consistency and reduces gradient batch-dissonance, it is unclear to what degree each property is responsible for the dramatic training improvements caused by including skip connections in deep networks. This thesis demonstrates several efficient methods by which each property can be measured concretely, which will enable a deeper understanding of why novel architectural traits benefit SGD and will allow more targeted exploration of the architecture space. Using two metrics we propose and several others from related literature, we also demonstrate that the amount of variance resulting from gradient step-inconsistency and batch-dissonance during SGD can be handled independently in an informed and targeted way by modifying the architecture or changing the learning-rate and batch-size parameters.

Chapter 1. Gradient Step-Consistency and Batch-Dissonance

When training a deep neural network using *Stochastic Gradient Descent* (SGD), high variance between gradient estimates relative to their magnitude causes instability and can prevent convergence. This variance can be separated into two distinct sources, which we will call *gradient step-inconsistency* and *gradient batch-dissonance*.

Definition 1.1 (**Gradient Step-Consistency**)**.** *Gradient step-consistency* is the degree to which a network state exhibits stability in the gradient when a small change is made to the network weights, such as during a step in gradient descent. For an architecture or network, gradient step-consistency refers to the degree to which network states typically experienced within the given reference frame (such as near the beginning or end of training while using industry-standard training methods) exhibit this same property. When referring to a lack of gradient step-consistency the phrase *gradient step-inconsistency* may be used.

Definition 1.2 (**Gradient Batch-Dissonance**)**.** *Gradient batch-dissonance* is the degree to which a network state exhibits disagreement between gradients calculated with different

inputs at the same network state. During stochastic gradient descent, gradients are calculated for a batch of inputs at each network state, hence the name gradient batch-dissonance. For an architecture or network, gradient batch-dissonance refers to the degree to which network states typically experienced under the same reference frame and with the same training set exhibit this property.

Networks with poor *gradient step-consistency* exhibit large relative variance between gradient estimates made before and after small steps in weight space. This is indicative of a rough loss surface and has been linked to poor convergence [12]. Networks with high *gradient batch-dissonance* have large relative variance between individual gradient estimates within each batch. This indicates that the batch-averaged gradient will reduce the loss less in each individual case and will be more likely to increase the loss for some training examples [3, 4, 18]. Poor gradient step-consistency and gradient batch-dissonance both contribute to the overall variance experienced between adjacent steps in SGD because the inputs and weights both change between each step.

The variance between adjacent steps in SGD can be controlled using the *batch-size* and *step-size* parameters. Namely, variance caused by gradient batch-dissonance can be reduced by increasing the batch-size and variance caused by gradient step-inconsistency can be reduced by decreasing the step-size. However, when gradient batch-dissonance is too high or gradient step-consistency is too low, convergence will not occur regardless of the step-size or batch-size chosen (see sections 2.1 and 3.1).

The terms gradient step-consistency and gradient batch-dissonance as we have defined them make it easier to talk about performance differences between different architectures, but they are not easy to measure directly. Several recent papers have defined concrete metrics to measure phenomena related to each term, but the lack of common terminology between these related papers has caused a degree of confusion and makes it more difficult to search the literature for papers on to each subject. Chapter 2: Gradient Step-Consistency and Chapter 3: Gradient Batch-Dissonance each contain several sections reviewing and validating results

for papers on each subject, and each outlines precisely how those papers fall under the topic of gradient step-consistency or gradient batch-dissonance as we have defined them.

We will demonstrate through our own results, and by those of the papers we review on each topic, that gradient step-consistency and gradient batch-dissonance both have a significant impact on the effectiveness and speed of SGD at finding a minimum for a given network. Because training performance is expensive to evaluate, is not deterministic, and can be impacted by thousands of options in architectural design and hyperparameter usage, optimizing for best training performance in an undirected way over any meaningful architecture space is impossible. By presenting, comparing, and analyzing metrics that measure gradient step-consistency and batch-dissonance efficiently, we hope that we can improve the discussion around the impact architectural design has on training and enable cheaper and more targeted searches for new architectural traits. Furthermore, we will investigate the connection between metrics on gradient step-consistency and batch-dissonance and the hyperparameters used to regulate their impact, step-size and batch-size, to determine if efficient optimal control of these hyperparameters might be possible.

1.0.1 Summary of Contents. The majority of this thesis will be dedicated to definitions and analysis of the various metrics that have been proposed to measure gradient step-consistency and batch-dissonance. In order to motivate these more clearly, we will begin with a more formal statement of our objectives, and outline the methods we will use accomplish them (section 1.1). In sections 1.2 and 1.3 we will define our terminology and notation. Thereafter, since the metrics we will introduce in chapters 2 and 3 are both strongly motivated by the *relative condition number*, we will briefly define *conditioning* and the relative condition number in section 1.4 and demonstrate how it can be efficiently estimated in section 1.5. The last section, 1.6, explains how random masking of gradient vectors can be used to dramatically decrease the temporal and spatial complexity of calculating the scale-invariant metrics we present.

Chapters 2 and 3 will consider gradient step-consistency and batch-dissonance respec-

tively in the mathematical context of conditioning including presentation of our own novel metrics and analysis of other metrics and results already present in the literature. Furthermore, these chapters will consider the effect of the step-size and batch-size parameters on reducing negative effects of high gradient step-inconsistency and batch-dissonance.

Chapter 4 will consist primarily of discussion on remaining objectives and the possible paths to achieve them, in addition to a brief conclusion section.

1.1 Objectives

Our objectives in publishing this thesis have been stated in a few different ways throughout the abstract and introduction. For clarity however, we will state them in a more formal way and then consider in a more detail how we will accomplish them. Our objectives are:

1. Summarize previous findings related to gradient step-consistency and batch-dissonance and place them in the uniform context of gradient conditioning.

2. Demonstrate efficient methods by which gradient step-consistency and batch-dissonance can be measured independently.

3. Investigate the impact gradient step-consistency and batch-dissonance each have on the stability and speed of SGD.

4. If possible, demonstrate that variance and instability from each source can be reduced or mitigated independently and in a targeted way using architectural or hyperparameter adjustments.

The remainder of this chapter will be dedicated to outlining the necessary context, proofs, and techniques that will be needed to accomplish the first two objectives. Then these will actually be accomplished in Chapter 2: Gradient Step Consistency and Chapter 3: Gradient Batch-Dissonance.

In the field of deep learning, when empirically evaluating novel architectures, techniques, metrics, etc. it is common practice to use the final performance or training speed of common

architectures as points of comparison. These are used to demonstrate some correlation or relationship between the novel content and the desired result. This will also be our approach to accomplishing objective 3 in Chapter 4. However, if gradient step-consistency and batch-dissonance are too strongly correlated in all of the architectures we test, it may be difficult to establish the independence between them that is required to accomplish objective 4. In other words, if no architectural traits we test appear to benefit gradient step-consistency or gradient batch-dissonance without also benefiting the other in a similar degree, we won't be able to test if targeted architectural modifications can alleviate instability caused by one or the other specifically.

1.2 Terminology

We will now proceed to more carefully define the terminology and notation we will use to define metrics on gradient step-consistency and batch-dissonance in the following chapters. In order to more easily discuss the plethora of factors which influence learning in Artificial Neural Networks, we will divide them into four families: namely the *objective*, the *architecture*, the *optimization*, and the *training set*.

The *objective* determines the ideal final state of a network, and its desired behavior. We will treat the loss function as the ideal numerical expression of that goal. In other words, throughout this paper we will assume that the loss function is absolute, or that minimizing it will result in exactly the desired behavior. All of the factors of interest then come from the following two categories: architecture and optimization.

We use the term *architecture* to refer to every part of our neural network which is determined before training begins. This includes the number and nature of layers, the type of activation function(s) used, the use of batch normalization, etc. Given an objective and dataset, these factors determine what the loss surface looks like when the full dataset is used to calculate the gradient, including how deep the desired minima are, and how easily they can be found using standard optimization techniques.

We use the term *optimizer*, or optimizer parameters, to refer to any and all user input to the neural network after training has begun. This includes the type of algorithm used (ADAM vs SGD etc.), learning rate schedules, momentum terms, etc. The chosen initialization is also part of the optimization, however, some architectures are designed with a specific method of sampling the initial weights as a requirement. In this specific case we will refer to the chosen sampling method as part of the architecture.

The *training set* significantly impacts training, but we are primarily concerned with how a given architecture generalizes and responds to perturbations in the data set. Because of this we will generally ignore factors in the training set that can impact training, such as improper labels and how representative the data set is of the data the network will be exposed to during deployment.

1.3 Notation

Unless otherwise specified, all subsequent notation in this thesis will be used according to the following definition:

Definition 1.3 (**Notation**)**.** Let an architecture $g : ((\mathbb{R}^m, ||\cdot||_{\mathbb{W}}), (\mathbb{R}^k, ||\cdot||_{\mathbb{X}})) \to \mathbb{R}^j$ and a loss function $\mathcal{L} : (\mathbb{R}^j, \mathbb{R}^h) \to \mathbb{R}$ be given. The following shortened notation will be used for these spaces throughout this thesis:

- $\mathbb{W}_g := \mathbb{R}^m$ is the flattened weight space space with norm $||\cdot||_{\mathbb{W}_g}$ induced by an inner product $\langle\cdot,\cdot\rangle_{\mathbb{W}_g}$
- $\mathbb{X}_g := \mathbb{R}^k$ is the flattened input space with norm $||\cdot||_{\mathbb{X}_g}$
- and $\mathbb{T}_g := \mathbb{R}^h$ is the space of target output values.

The subscript naming the architecture, g in this case, will be left off if an architecture has not been specified.

Given parameters $w \in \mathbb{W}$, $x \in \mathbb{X}$ and $t \in \mathbb{T}$, we will use the following shortened notation for the loss and gradient functions:

- for the loss:

$$\mathcal{L}_g(w, x, t) := (\mathcal{L} \circ g)(w, x, t)$$

- for the gradient of the loss with respect to the weight space:

$$h_{\mathbb{W}}(w, x, t) := \nabla_{\mathbb{W}} \mathcal{L}_g(w, x, t)$$

- and for the gradient of the loss with respect to the input space:

$$h_{\mathbb{X}}(w, x, t) := \nabla_{\mathbb{X}} \mathcal{L}_g(w, x, t) \ .$$

When a batch of inputs is used instead of individual inputs we will use the following notation:

- the set of all batches of size b (the subscript b will be left off if a batch size has not been specified):

$$B_b(\mathbb{X}) := \{(x_i, t_i) \mid 0 < i \leq n, \ (x_i, t_i) \in \mathbb{X} \times \mathbb{T}\}$$

- a batch of gradient estimates:

$$h_{\mathbb{W}}(w, \chi) := \{h_{\mathbb{W}}(w, x_i, t_i) \mid (x_i, t_i) \in \chi\}, \ \chi \in B_b(\mathbb{X})$$

- the average of a batch of gradient estimates:

$$\overline{h_{\mathbb{W}}(w, \chi)} := \frac{1}{b} \sum_{(x_i, t_i) \in \chi} h_{\mathbb{W}}(w, x_i, t_i), \ \chi \in B_b(\mathbb{X})$$

- and for the true gradient (averaged over the entire dataset or theoretical sample space):

$$h_{\mathbb{W}}(w) \ .$$

1.4 What is Conditioning and How is it Related to SGD?

A number of metrics and (expensive) visualizations have been proposed that are closely related to gradient step-consistency and gradient batch-dissonance as we have defined the terms. A significant number of papers have investigated how various architectural traits impact the roughness of the loss surface, which is strongly related to gradient step-consistency as we have defined it (see section 2.4). Recently, several metrics have also been proposed for measuring gradient batch-dissonance (see section 3.4). Using these metrics, a few architectural traits have been shown to slow or prevent increases in gradient step-inconsistency and batch-dissonance caused by increasing network depth. These traits enable the training of much deeper networks and can make training significantly cheaper by increasing the range of viable batch-sizes and learning rates. The most well-known architectural trait that does this is the *skip connection* (subsection 2.4.2).

For example, architectures with skip connections have been shown to exhibit much smaller variance between gradient estimates at small step-sizes than networks without skip connections [12]. Furthermore, skip connections have been shown to significantly reduce variance between gradient estimates made with batch elements that are similar [1]. Thus skip connections decrease gradient variance caused by differences in both the input and weights. In other words, they improve the *conditioning* of the network gradient function with respect to both the weights and inputs, thereby decreasing both gradient step-inconsistency and batch-dissonance.

By measuring the *relative conditioning* of the gradient function (Definition 1.5) as opposed to variance or relative variance, we can more directly measure the effect architectural attributes like skip connections have on gradient step-consistency and gradient batch-dissonance. Furthermore, measuring the conditioning of an architecture with respect to weights and the inputs individually may allow for informed and targeted architectural or hyperparameter changes in order to improve the conditioning or reduce the effect the resulting variance has on the speed and stability of SGD (see sections 2.1 and 3.1).

In order to motivate further discussion of conditioning and its relation to gradient step-consistency and batch-dissonance, we will proceed with brief definitions and explanations of the absolute and relative condition numbers. We will begin by defining the absolute condition number, which is motivated by the Fréchet derivative.

Definition 1.4 (**Absolute Condition Number**). Given a function $f : (\mathbb{R}^n, ||\cdot||_\alpha) \to (\mathbb{R}^m, ||\cdot||_\beta)$, the absolute condition number $\hat{\kappa}$ at input $x \in \mathbb{R}^n$ is defined as follows:

$$\hat{\kappa}_f(x) = \lim_{\epsilon \to 0} \sup_{||h||_\alpha < \epsilon} \frac{||f(x+h) - f(x)||_\beta}{||h||_\alpha} = ||J_f(x)||_{\alpha,\beta} \tag{1.1}$$

where $h \in \mathbb{R}^n$, $J_f(x)$ is the Jacobian of f at x, and $||\cdot||_{\alpha,\beta}$ is the induced norm. The second equality holds if and only if $J_f(x)$ exists, which is guaranteed if f is differentiable at x.

The absolute condition number measures the largest possible change to the output given a measured perturbation in the input. In other words, it is a measure of how sensitive a function is to changes or error in the input. Small changes in input are unavoidable in floating point implementations, and the condition number indicates how much these errors can be magnified in the output.

To put this in perspective, consider the function $g(x) = e^x$. Near $x = 1$, e^x is very well conditioned. In a 32 bit floating point system machine epsilon ($\epsilon_{m|x=1}$) is approximately $1.2 * 10^{-7}$ at $x = 1$, and a change of $\frac{1}{2}\epsilon_{m|x=1}$ in the input results in a change in the output of $1.6 * 10^{-7} \approx .67 * \epsilon_{m|x=2.7}$. In this case $\hat{\kappa}_g(1) \approx 2.7$. However, e^x isn't as well conditioned for higher values of x. At $x = 50$ a change of $\frac{1}{2}\epsilon_{m|x=50}$ $(1.9 * 10{-}6)$, which is slightly larger now because floating point systems lose precision further away from zero, results in a change in the output of $1.9 * 10^{16}$. Now $\hat{\kappa}_g(1) \approx 5.2 * 10^{21}$. Changing from an input with at least six accurate decimal points to an output that could be off by as much as $1.9{*}10^{16}$ is a massive loss in precision, and that is accurately represented in the absolute condition number. However, $e^{50} \approx 5.18 * 10^{21}$, meaning that the first five digits of the answer are in fact correct, and we only lost two digits of accuracy. Since $\epsilon_{m|x=e^{50}} \approx 5.7 * 10^{14}$, the total error is only about 17.5

times machine epsilon.

Because e^{50} is so large in our example, the absolute condition number made the loss in accuracy appear much worse than it actually was. Without access to the size of the output we would have had no way to judge how bad a precision loss of $1.9 * 10^{16}$ was relative to machine epsilon at our output. Because of this issue a normalized version of the condition number, known as the relative condition number, is much more commonly used.

Definition 1.5 (**Relative Condition Number**). Given a function $f : (\mathbb{R}^n, ||\cdot||_\alpha) \to (\mathbb{R}^m, ||\cdot||_\beta)$, the relative condition number κ at $x \in (\mathbb{R}^n, ||\cdot||_\alpha)$ is defined as follows:

$$\kappa_f(x) = \lim_{\epsilon \to 0} \sup_{||h||_\alpha < \epsilon} \frac{||f(x+h) - f(x)||_\beta}{||h||_\alpha} \cdot \frac{||x||_\alpha}{||f(x)||_\beta} = \frac{||J_f(x)||_{\alpha,\beta} \cdot ||x||_\alpha}{||f(x)||_\beta} \tag{1.2}$$

where $h \in \mathbb{R}^n$, $J_f(x)$ is the Jacobian of f at x, and $||\cdot||_{\alpha,\beta}$ is the induced norm. The second equality holds if and only if $J_f(x)$ exists, which is guaranteed if f is differentiable at x.

Essentially, the relative condition number is just the absolute condition number normalized by the relative scale of the output and input. In our previous example $g(x) = e^x$, the relative condition number at $x = 1$ is in fact exactly 1, and at $x = 50$ the relative condition number is 50. That's because in $\mathbb{R}^1$ the absolute condition number is simply the magnitude of the derivative, and of course $|\frac{d}{dx}e^x| = |e^x|$. Once we normalize by $|\frac{x}{e^x}|$ we just get $|x|$ back. In any case, the relative condition number is much more informative on its own than the absolute condition number. As a general rule of thumb, if $\kappa_f(x) \approx 10^k$, then up to about k digits of accuracy can be lost.

1.4.1 Simple Example Network.

Deep neural networks contain dozens of nested calculations and can therefore become extremely poorly conditioned because of seemingly simple decisions, such as the choice of activation function. Now that we've defined what relative conditioning is, we will demonstrate how this poor conditioning can occur easily in the gradient of a deep neural network with a simple example.

Consider a small network with two hidden layers and an output layer, each with weight W_i and bias b_i. The activation function is f and the input to the network is x. This network can be expressed in a system of equations as follows:

$$\begin{aligned} a_1 &= W_1 x \ + b_1, \quad h_1 = f(a_1) \ , \\ a_2 &= W_2 h_1 + b_2, \quad h_2 = f(a_2) \ , \\ a_3 &= W_3 h_2 + b_3, \text{ and } \hat{y} \ = f_y(a_3) \ . \end{aligned} \tag{1.3}$$

For simplicity we choose f_y to be a logistic activation function and use binary cross-entropy loss as our loss function $\mathcal{L}_e$:

$$\mathcal{L}_e = -\frac{1}{N} \sum_{m,n} \log \hat{y}_{mn} + (1 - y_{mn}) \log(1 - \hat{y}_{mn}) \ . \tag{1.4}$$

This results in the following gradients and weight update rule:

$$\begin{aligned} \delta a_3 &= \tfrac{\partial \mathcal{L}_e}{\partial a_3} = \hat{y} - y, & \delta W_3 &= -\delta a_3 h_2^T, \\ \delta a_2 &= \tfrac{\partial \mathcal{L}_e}{\partial a_2} = (W_3^T \delta a_3) \odot f'(a_2), & \delta W_2 &= -\delta a_2 h_1^T, \\ \delta a_1 &= \tfrac{\partial \mathcal{L}_e}{\partial a_1} = (W_2^T \delta a_2) \odot f'(a_1), & \delta W_1 &= -\delta a_1 x^T, \\ & & W_{i,n} &= W_{i,n-1} + lr \cdot \delta W_{i,n-1} \end{aligned} \tag{1.5}$$

where $\odot$ is the element-wise product and lr is the *learning rate*, which is a hyperparameter used control the step-size in SGD.

1.4.2 Example: Poor Conditioning caused by Activation Functions. For our first motivating example, we will consider how poor conditioning in the gradient function with respect to the input space can be caused by the choice of activation function.

The derivative of the common activation function ReLU contains a discontinuity. By the chain rule we know that the derivative of the activation function will appear in every element of the gradient for layers it is used on, evaluated on the value a_i from the forward pass (see δW_i, $i < n$ in 1.5). When f' has a discontinuity it introduces a discontinuity in δa_i

for every element of a_i, which is directly dependent on the input to the network (x in 1.3). In other words, in a network where every layer is followed by a ReLU activation function there will be as many jump discontinuities in the gradient function as there are neurons in the network. Furthermore, these jump discontinuities are concentrated in the region of space ($\mathbb{W} \times \mathbb{X}$) where the network performs at full capacity, because the discontinuity is located at the boundary between the active and inactive state for that neuron in the forward pass.

A number of different activation functions have been introduced that seek to avoid this issue. Some, including GELU, SiLU, Softplus, and ELU, remove the discontinuity by smoothly approximating ReLU. Unfortunately, these have not demonstrated a clear ability to increase viable training depth.

Another activation function, CReLU [20], has been used in conjunction with symmetric block initialization to minimize jump discontinuities and behave linearly at initialization [1]. Balduzzi et al. demonstrated that this combination showed significantly improved conditioning in the derivative of the loss function (and therefore in the network gradient as well) with respect to the input. Furthermore, they successfully trained a 200 layer fully-connected network that outperformed a 200 layer ResNet! This result is astonishing because fully connected networks with other common activation functions and initializations become impossible to train effectively at much lower depths.

After the first step the network must diverge partially from this linear behavior, but the introduced jump discontinuities are significantly smaller than those in the derivative than ReLU in regions of the weight space close to the initialization point (see figure 3.11). This activation-function initialization pairing will be revisited in section 3.4.

1.5 Estimating Gradient Conditioning

Because it is so expensive to evaluate the performance of large architectures it is infeasible to systematically search for new architectures, even with very simple extrapolations from currently known architectural styles. This is why we would like to develop efficient metrics

that allow for cheaper and more systematic evaluation of the impact of novel architectural traits on gradient stability. We believe that the most natural starting point in developing these metrics is in the conditioning of the gradient function, which is mathematically very simple but equally meaningful. We will begin by introducing a theorem that can be used to efficiently estimate the relative condition number. Subsequently, we will define precisely how the conditioning of a gradient estimation with respect to the weight space and the input space can be measured independently.

Theorem 1.6. *Let a function $f : (\mathbb{R}^n, ||\cdot||_\alpha) \to (\mathbb{R}^m, ||\cdot||_\beta)$ and a point $x \in (\mathbb{R}^n, ||\cdot||_\alpha)$ be given. Let the epsilon ball around x, $B_\epsilon(x)$, be defined as follows:*

$$B_\epsilon(x) := \{y \in (\mathbb{R}^n, ||\cdot||_\alpha) : ||y - x||_\alpha < \epsilon\} \ . \tag{1.6}$$

If f is differentiable at x, then an $\epsilon > 0$ exists for which every $p \in B_\epsilon(x) \setminus \{x\}$ satisfies the following relation:

$$\kappa_f(x) + o(\epsilon) \geq \frac{||f(p) - f(x)||_\beta ||x||_\alpha}{||p - x||_\alpha ||f(x)||_\beta} \ . \tag{1.7}$$

Proof: Assume the hypothesis. Since f is differentiable at x, we can estimate $f(p)$ using a linear approximation as follows:

$$f(p) = f(x) + J_f(x)(p - x) + \delta \ ,$$

where the error $||\delta||_\beta = ||f(p) - f(x) - J_f(x)(p - x)||_\beta \in o(||p - x||_\alpha)$. In other words, $||\delta||_\beta \to 0$ as $p \to x$ significantly faster than $||x - p|| \to 0$.

Thus by the properties of the induced norm $||\cdot||_{\alpha,\beta}$, we can establish the bound

$$\frac{||J_f(x)(p - x)||_\beta}{||p - x||_\alpha} \leq ||J_f(x)||_{\alpha,\beta} \ ,$$

and, since f is differentiable at x:

$$\kappa_f(x) = \frac{||J_f(x)||_{\alpha,\beta} \cdot ||x||_\alpha}{||f(x)||_\beta} \geq \frac{||J_f(x)(p-x)||_\beta ||x||_\alpha}{||p-x||_\alpha ||f(x)||_\beta} = \frac{||f(p)-f(x)-\delta||_\beta ||x||_\alpha}{||p-x||_\alpha ||f(x)||_\beta} \ .$$

Furthermore, by the reverse triangle inequality:

$$\begin{aligned}
\frac{||f(p)-f(x)-\delta||_\beta ||x||_\alpha}{||p-x||_\alpha ||f(x)||_\beta} &\geq \left| \left(\frac{||f(p)-f(x)||_\beta}{||p-x||_\alpha} - \frac{||\delta||_\beta}{||p-x||_\alpha} \right) \frac{||x||_\alpha}{||f(x)||_\beta} \right| \\
\implies \kappa_f(x) &\geq \left| \frac{||f(p)-f(x)||_\beta ||x||_\alpha}{||p-x||_\alpha ||f(x)||_\beta} - \frac{||\delta||_\beta ||x||_\alpha}{||p-x||_\alpha ||f(x)||_\beta} \right| \\
\implies \kappa_f(x) + \phi &\geq \frac{||f(p)-f(x)||_\beta ||x||_\alpha}{||p-x||_\alpha ||f(x)||_\beta} \text{ where } \phi = \frac{||\delta||_\beta ||x||_\alpha}{||p-x||_\alpha ||f(x)||_\beta} \ .
\end{aligned}$$

Because $||\delta||_\beta \in o(||p-x||_\alpha)$ we have

$$\frac{||\delta||_\beta ||x||_\alpha}{||p-x||_\alpha ||f(x)||_\beta} \in o(\epsilon) \quad \forall p \in B_\epsilon(x) \ ,$$

meaning that

$$\kappa_f(x) + o(\epsilon) \geq \frac{||f(p)-f(x)||_\beta ||x||_\alpha}{||p-x||_\alpha ||f(x)||_\beta} \tag{1.7}$$

for any $p \in B_\epsilon(x) \setminus \{x\}$ with sufficiently small ϵ. □

Using this relationship we can efficiently estimate a lower bound on the relative condition number of a function. Furthermore, if we have a set of n samples, we can improve our estimate of κ_f simply by taking the maximum over all samples.

Definition 1.7 (Approximate Best Lower Bound on κ)**.** Let $\epsilon > 0$ and $n \in \mathbb{N}$ be given. Then we define our approximate best lower bound on κ as

$$\kappa_f^{\gtrapprox}(x) := \max_{p_i \in B_\epsilon(x), 0<i\leq n} \frac{||f(p_i)-f(x)||_\beta ||x||_\alpha}{||p_i-x||_\alpha ||f(x)||_\beta} \ . \tag{1.8}$$

1.6 Reducing Metric Costs with Random Masking

Unfortunately the memory requirement to calculate some of the metrics we will introduce in chapters 2 and 3 can be quite large, depending on the batch-size and number of gradient samples drawn. However, because random mappings on large vectors approximately preserve distance and inner products between vectors [9], we can significantly reduce the temporal and spatial complexity of calculating these metrics by applying the same random mapping to each gradient vector prior to metric calculation.

We apply some constraints to the mask generation process. We require that each layer of the network is sampled evenly relative to the number of parameters it contains. Thus when we refer to a 5% mask we mean a random masking where 5% of the gradient vector is chosen from each layer and stacked. The number of samples chosen from each layer is always rounded up, so every layer will have at least one element present in each mask. The mask is generated when the network is initialized and used consistently throughout the training process.

Each time a metric is introduced by ourselves or is included in a review of another paper, a figure will be shown documenting the effects of random masking on that metric. This figure will include calculated average absolute error and computation time relative to the unmasked metric for a ResNet with 26 layers and about 7,500,000 parameters. In each case the error and timing is averaged over samples from five short training runs. Three masks of each size were used in each training run, with samples drawn evenly throughout five training epochs on CIFAR10. Our mask implementation is not parallelized and could be improved significantly. In some cases, in particular where metrics require several different mask calls during evaluation and are inexpensive to calculate once all inputs are provided, it can be temporally cheaper to not use a mask. When this is the case, the caption of the mask result and timing figure for that metric will make note if it.

Chapter 2. Gradient Step-Consistency

Since introducing the term gradient step-consistency in the first chapter we have discussed several other topics, so we will review the term before proceeding. Recall, from Definition 1.1, that gradient step-consistency is a quality of networks and architectures, namely that of having stable gradients across small steps in the weight space. Architectures that have high gradient step-consistency can be trained much more quickly and efficiently using techniques such as momentum. Furthermore the actual decrease in loss from a given step will be closer to the expected decrease in loss (i.e., the derivative of the loss function in the direction of the step times the step size). In other words, larger step sizes can be used without unexpected changes in the loss occurring and the expected gain from a step of fixed size is higher. Before moving on however, we'll establish these in a theorem and proof.

2.1 Quantifying Gradient Step-Consistency

Given two neural networks, if one network exhibits higher gradient step-consistency than the other within a provided context, then there are several quantifiable performance differences that will be true in expectation within that context. In order to demonstrate these, however, we need a more concrete mathematical definition of gradient step consistency.

Definition 2.1 (**A Formal Definition of Gradient Step-Consistency**)**.** Let two neural network architectures be given, one of which (network A) exhibits higher gradient step-consistency than the other (network B) within some context (including a shared dataset $\mathbb{X} = \mathbb{Y}$ or two datasets $\mathbb{X} \neq \mathbb{Y}$). We will more formally define the relationship implied by this statement as follows:

For any regions $\Phi \in (\mathbb{W}_A, \|\cdot\|_A)$ and $\Psi \in (\mathbb{W}_B, \|\cdot\|_B)$, and batches $\chi \in B(\mathbb{X})$, $\gamma \in B(\mathbb{Y})$, that are contained within the provided context, there exists an $\epsilon > 0$ such that the following

relationship holds for any δ_1, δ_2 between zero and ϵ:

$$\begin{aligned} &\mathop{\mathbb{E}}_{\phi\in\Phi,\chi\in B(\mathbb{X})}\left[\frac{\left\|\overline{h_{\mathbb{W}_A}(\phi,\chi)}-\overline{h_{\mathbb{W}_A}(\phi-\alpha,\chi)}\right\|_A}{\|\alpha\|_A}\right] \\ <&\mathop{\mathbb{E}}_{\psi\in\Psi,\gamma\in B(\mathbb{Y})}\left[\frac{\left\|\overline{h_{\mathbb{W}_B}(\psi,\gamma)}-\overline{h_{\mathbb{W}_B}(\psi-\beta,\gamma)}\right\|_B}{\|\beta\|_B}\right] \\ &\text{where}\ \ \alpha\ =\ \delta_1\ \overline{h_{\mathbb{W}_A}(\phi,\chi)}\ , \\ &\text{and}\ \ \beta\ =\ \delta_2\ \overline{h_{\mathbb{W}_B}(\psi,\gamma)}\ . \end{aligned} \tag{2.1}$$

There are a few ways in which networks with high gradient step-consistency demonstrate superior performance during learning compared to similar networks with lower gradient step-consistency. In order to prove some of these benefits, the following lemma will be necessary.

Lemma 2.2. *Given positive scalar values $\delta_A, \delta_B \in \mathbb{R}$, and vectors $u, v \in (\mathbb{R}^n, \|\cdot\|_A)$ and vectors $x, y \in (\mathbb{R}^m, \|\cdot\|_B)$, where*

$$\frac{\|u-v\|_A}{\|\delta_A u\|_A} < \frac{\|x-y\|_B}{\|\delta_B x\|_B}\ , \tag{2.2}$$

and $\|\cdot\|_A$ and $\|\cdot\|_B$ are each induced by an inner product, we have that

$$\frac{\|u\|_A^2-\langle u,v\rangle_A}{\|\delta_A u\|_A^2} < \frac{\|x\|_B^2-\langle x,y\rangle_B}{\|\delta_B y\|_B^2}+\epsilon+\epsilon^2 \tag{2.3}$$

with

$$0<\epsilon\in\mathcal{O}\left(\frac{\|x-y\|_B}{\|\delta_B x\|_B}\right)\ .$$

More specifically, the tightest bound that can be established using inequality (2.2) is the fol-

lowing:

$$\left(\frac{\|u\|_A^2 - \langle u, v\rangle_A}{\|\delta_A u\|_A^2}\right) < \left(\frac{\|x\|_B^2 - \langle x, y\rangle_B}{\|\delta_B x\|_B^2}\right) + \frac{1}{2}\left(\left(\frac{\|y\|^2}{\|\delta_B x\|^2} - \frac{\|v\|^2}{\|\delta_A u\|^2}\right) + (\epsilon_2 - \epsilon_1)\right) \tag{2.4}$$

$$\textit{where} \quad \epsilon_1 = \frac{\|u - v\|_A}{\|\delta_A u\|_A} \quad \textit{and} \quad \epsilon_2 = \frac{\|x - y\|_B}{\|\delta_B x\|_B} \ .$$

Furthermore, if

$$\epsilon_1 - \frac{\|v\|_A}{\|\delta_A u\|_A} < \epsilon_2 - \frac{\|y\|_B}{\|\delta_B x\|_B} \ , \tag{2.5}$$

inequality (2.3) holds even with the assumption that $\epsilon = 0$.

Finally, in the case that

$$\frac{\|v\|_A}{\|\delta_A u\|_A} = \frac{\|y\|_B}{\|\delta_B x\|_B} \ ,$$

the width of the distance between the primary terms in inequality (2.4) is equal to one-half of the width of the gap in inequality (2.2) and is in the same direction. In this case we also have that $\theta_{u,v}$ *and* $\theta_{x,y}$ *can each be expressed as a value between* 0 *and* π *inclusive where* $\theta_{u,v} < \theta_{x,y}$.

Proof. Assume the hypothesis. Using properties of inner products on a real space we have that

$$\begin{aligned}
\frac{\|u - v\|_A^2}{\|\delta_A u\|_A^2} &= \frac{\langle u, u - v\rangle_A + \langle v, v - u\rangle_A}{\delta_A \|u\|^2} \\
&= \left(\frac{\langle u, u - v\rangle_A}{\delta_A \|u\|_A^2}\right)_{1A} + \left(\frac{\langle v, v - u\rangle_A}{\delta_A \|u\|_A^2}\right)_{2A} \\
&= \left(\frac{\|u\|_A^2 - \langle u, v\rangle_A}{\delta_A \|u\|_A^2}\right)_{1A} + \left(\frac{\|v\|_A^2 - \langle u, v\rangle_A}{\delta_A \|u\|_A^2}\right)_{2A}
\end{aligned} \tag{2.6}$$

where we have separated the terms into groups 1 and 2 for ease of reference moving forward. No terms will be transferred between these two groups and the letters A and B will be used

to designate whether a group corresponds to the left or right sides of inequality (2.2).

In order to establish a bound on the magnitude of $((\cdot)_{2B} - (\cdot)_{2A})$ in terms of the known quantities ϵ_1, ϵ_2, or $\epsilon_2 - \epsilon_1$ (the gap in inequality (2.2)), we need to break each group of terms down into quantities that are directly comparable to them. We accomplish this by replacing the inner product in group A with an expression derived using the law of cosines:

$$
\begin{aligned}
\left(\frac{\|v\|_A^2 - \langle u, v\rangle}{\delta_A \|u\|_A^2}\right)_{2A} &= \frac{1}{\delta_A}\left(\frac{\|v\|_A^2}{\|u\|_A^2} - \frac{\|u\|_A \|v\|_A \cos(\theta_{u,v})}{\|u\|_A^2}\right) \\
&= \frac{1}{\delta_A}\left(\frac{\|v\|_A^2}{\|u\|_A^2} - \frac{\|v\|_A}{\|u\|_A}\left(\frac{\|u\|_A^2 + \|v\|_A^2 - \|u-v\|_A^2}{2\|u\|_A \|v\|_A}\right)\right) \\
&= \frac{1}{\delta_A}\left(\frac{\|v\|_A^2}{\|u\|_A^2} - \frac{\|v\|_A}{\|u\|_A}\left(\frac{\|u\|_A}{2\|v\|_A} + \frac{\|v\|_A}{2\|u\|_A} - \frac{\|u-v\|_A^2}{2\|u\|_A\|v\|_A}\right)\right) \\
&= \frac{1}{\delta_A}\left(\frac{\|v\|_A^2}{\|u\|_A^2} - \frac{1}{2} - \frac{\|v\|_A^2}{2\|u\|_A^2} + \frac{\|u-v\|_A^2}{2\|u\|_A^2}\right) \\
&= \frac{1}{2\delta_A}\left(\frac{\|v\|_A^2}{\|u\|_A^2} - 1\right) + \frac{\|u-v\|_A^2}{2\delta_A\|u\|_A^2} \qquad (2.7) \\
&= \frac{1}{2\delta_A}\left(\left(\frac{\|v\|_A - \|u\|_A}{\|u\|_A} + 1\right)^2 - 1\right) + \frac{\|u-v\|_A^2}{2\delta_A\|u\|_A^2} \\
&= \frac{1}{2\delta_A}\left(\left(\frac{\|v\|_A - \|u\|_A}{\|u\|_A}\right)^2 + 2\frac{\|v\|_A - \|u\|_A}{\|u\|_A} + 1 - 1\right) + \frac{\|u-v\|_A^2}{2\delta_A\|u\|_A^2} \\
&= \frac{1}{2}\left(\frac{\|v\|_A - \|u\|_A}{\sqrt{\delta_A}\|u\|_A}\right)^2 + \frac{\|v\|_A - \|u\|_A}{\delta_A\|u\|_A} + \frac{\|u-v\|_A^2}{2\|\delta_A u\|_A^2} \\
&= \frac{\epsilon_{1^*}^2}{2} + \frac{\epsilon_{1^*}}{\sqrt{\delta_A}} + \frac{\epsilon_1^2}{2} \\
&\text{where } \epsilon_1 = \frac{\|v-u\|_A}{\|\delta_A u\|_A} \text{ and } \epsilon_{1^*} = \frac{\|v\|_A - \|u\|_A}{\|\delta_A u\|_A} .
\end{aligned}
$$

By the same logic

$$
\left(\frac{\|y\|_B^2 - \langle x, y\rangle_B}{\delta_B \|x\|_B^2}\right)_{2B} = \frac{\epsilon_{2^*}^2}{2} + \frac{\epsilon_{2^*}}{\sqrt{\delta_B}} + \frac{\epsilon_2^2}{2} \text{ where } \epsilon_2 = \frac{\|y-x\|_B}{\|\delta_B x\|_B} \text{ and } \epsilon_{2^*} = \frac{\|y\|_B - \|x\|_B}{\|\delta_B x\|_B} .
$$

Furthermore, we know by the reverse triangle inequality that

$$|\epsilon_{1^*}| \leq \epsilon_1 \text{ and } |\epsilon_{2^*}| \leq \epsilon_2 \ . \tag{2.8}$$

Because z^2 is a monotonically increasing function for any positive $z \in \mathbb{R}$ we have that

$$\frac{\|u-v\|_A}{\|\delta_A u\|_A} < \frac{\|x-y\|_B}{\|\delta_B x\|_B} \ , \tag{2.9}$$

which implies that

$$\begin{aligned} \frac{\|u-v\|_A^2}{\|\delta_A u\|_A^2} &= \left(\frac{\|u\|_A^2 - \langle u,v\rangle_A}{\|\delta_A u\|_A^2}\right)_{1A} + \left(\frac{\epsilon_{1^*}^2}{2} + \frac{\epsilon_{1^*}}{\sqrt{\delta_A}} + \frac{\epsilon_1^2}{2}\right)_{2A} \\ &< \left(\frac{\|x\|_B^2 - \langle x,y\rangle_B}{\|\delta_B x\|_B^2}\right)_{1B} + \left(\frac{\epsilon_{2^*}^2}{2} + \frac{\epsilon_{2^*}}{\sqrt{\delta_B}} + \frac{\epsilon_2^2}{2}\right)_{2B} = \frac{\|x-y\|_B^2}{\|\delta_B x\|_B^2} \ . \end{aligned}$$

This gives us that

$$\Longrightarrow$$

$$\left(\frac{\|u\|_A^2 - \langle u,v\rangle_A}{\|\delta_A u\|_A^2}\right) < \left(\frac{\|x\|_B^2 - \langle x,y\rangle_B}{\|\delta_B x\|_B^2}\right) + \left(\frac{\epsilon_{2^*}^2 - \epsilon_{1^*}^2}{2} + \frac{\epsilon_{2^*}}{\sqrt{\delta_B}} - \frac{\epsilon_{1^*}}{\sqrt{\delta_A}} + \frac{\epsilon_2^2 - \epsilon_1^2}{2}\right) \ . \tag{2.10}$$

Until this point in time we have not made any approximations at all, so the above expression is as tight an inequality as can be obtained using inequality (2.2). However, we can express the error in big-O notation using a few approximations.

We know that $0 \leq |\epsilon_{1^*}| < \epsilon_1$ and $0 \leq |\epsilon_{2^*}| \leq \epsilon_2$. Thus we also have that $|\epsilon_{2^*}^2 - \epsilon_{1^*}^2| \leq \epsilon_2^2$. These allow us to simplify the error term into an expression that depends only on the known

quantities ϵ_2 and $\epsilon_2^2 - \epsilon_1^2$:

$$\begin{aligned}\frac{\epsilon_{2^*}^2 - \epsilon_{1^*}^2}{2} + (\frac{\epsilon_{2^*}}{\sqrt{\delta_B}} - \frac{\epsilon_{1^*}}{\sqrt{\delta_A}}) + \frac{\epsilon_2^2 - \epsilon_1^2}{2} &\leq \left|\frac{\epsilon_{2^*}^2 - \epsilon_{1^*}^2}{2}\right| + \frac{|\epsilon_{2^*}|}{\sqrt{\delta_B}} + \frac{|\epsilon_{1^*}|}{\sqrt{\delta_A}} + \frac{\epsilon_2^2 - \epsilon_1^2}{2} \\ &\leq \frac{\epsilon_2^2}{2} + \frac{\epsilon_2}{\sqrt{\delta_B}} + \frac{\epsilon_1}{\sqrt{\delta_A}} + \frac{\epsilon_2^2 - \epsilon_1^2}{2} = \epsilon_2^2 + \left(\frac{1}{\sqrt{\delta_A}} + \frac{1}{\sqrt{\delta_B}}\right)\epsilon_2 \ ,\end{aligned}$$

which means that

$$\frac{\|u\|_A^2 - \langle u, v\rangle_A}{\|\delta_A u\|_A^2} < \frac{\|x\|_B^2 - \langle x, y\rangle_B}{\|\delta_B x\|_B^2} + \epsilon_2^2 + \left(\frac{1}{\sqrt{\delta_A}} + \frac{1}{\sqrt{\delta_B}}\right)\epsilon_2 \ .$$

Thus if $\epsilon_2 \leq 1$ the error is $\mathcal{O}(\epsilon_2)$, otherwise it is $\mathcal{O}(\epsilon_2^2)$.

In order to prove the final statements about the width of the gap in the above inequality, and to express it in a way that is easier to interpret without making any approximations, consider the intermediate expression 2.7 from earlier in this proof:

$$\begin{aligned}\left(\frac{\|v\|_A^2 - \langle u, v\rangle_A}{\|\delta_A u\|_A^2}\right)_{2A} &= \frac{1}{2\delta_A}\left(\frac{\|v\|_A^2}{\|u\|_A^2} - 1\right) + \frac{\|u - v\|_A^2}{2\|\delta_A u\|_A^2} \\ &= \frac{\|v\|_A^2}{2\|\delta_A u\|_A^2} - \frac{1}{2\delta_A} + \frac{\epsilon_1^2}{2} \ .\end{aligned}$$

This expression gives us the following:

$$\begin{aligned}&\left(\frac{\|u\|_A^2 - \langle u, v\rangle_A}{\|\delta_A u\|_A^2}\right)_{1A} + \left(\frac{\|v\|_A^2}{2\|\delta_A u\|_A^2} - \frac{1}{2\delta_A} + \frac{\epsilon_1^2}{2}\right)_{2A} \\ &< \left(\frac{\|x\|_B^2 - \langle x, y\rangle_B}{\|\delta_B x\|_B^2}\right)_{1B} + \left(\frac{\|y\|_B^2}{2\|\delta_B x\|_B^2} - \frac{1}{2\delta_B} + \frac{\epsilon_2^2}{2}\right)_{2B} \\ &\Longrightarrow \\ \left(\frac{\|u\|_A^2 - \langle u, v\rangle_A}{\|\delta_A u\|_A^2}\right)_{1A} &< \left(\frac{\|x\|_B^2 - \langle x, y\rangle_B}{\|\delta_B x\|_B^2}\right)_{1B} + \frac{1}{2}\left(\frac{\|y\|_B^2}{\|\delta_B x\|_B^2} - \frac{\|v\|_A^2}{\|\delta_A u\|_A^2}\right) \\ &\quad + \left(\frac{1}{\delta_A} - \frac{1}{\delta_B}\right) + \frac{\epsilon_2^2 - \epsilon_1^2}{2} \ ,\end{aligned} \tag{2.4}$$

as desired. Since no approximations or substitutions were made in the derivation, this bound

is equally as tight as in inequality (2.2). Furthermore, because $\epsilon_2^2 - \epsilon_1^2$ is the gap size in the squared version of inequality (2.2), we know that if

$$\frac{1}{2}\left(\frac{\|y\|_B^2}{\|x\|_B^2} - \frac{\|v\|_A^2}{\|u\|_A^2}\right) + \left(\frac{1}{\delta_A} - \frac{1}{\delta_B}\right) + \frac{\epsilon_2^2 - \epsilon_1^2}{2} < \epsilon_2^2 - \epsilon_1^2$$

or, equivalently, if

$$\epsilon_1 - \frac{\|v\|_A^2}{\|u\|_A^2} - \frac{1}{\delta_A} < \epsilon_2 - \frac{\|y\|_B^2}{\|x\|_B^2} - \frac{1}{\delta_B} \ ,$$

then inequality (2.3) will hold even when the error term ϵ is assumed to be zero.

In order to prove the final two points of the lemma, consider the case where

$$\frac{\|y\|_B}{\|x\|_B} = \frac{\|v\|_A}{\|u\|_A} \ \text{ and } \ \delta_A = \delta_B \ .$$

Then the only non-zero error term in inequality (2.4) is $\frac{\epsilon_2^2 - \epsilon_1^2}{2}$. We know that the width of the gap in the squared version of inequality (2.2) we used is $\epsilon_2^2 - \epsilon_1^2$ and is strictly positive. Therefore, $(\cdot)_{1B} - (\cdot)_{1A}$ is smaller than $\epsilon_2^2 - \epsilon_1^2$ by $\frac{\epsilon_2^2 - \epsilon_1^2}{2}$. In other words, excluding the error term in this case cuts the width of the gap precisely in half while preserving the original direction of the inequality.

Furthermore, by replacing each inner product with $\|u\|_A \|v\|_A \cos(\theta_{u,v})$ or $\|x\|_B \|y\|_B \cos(\theta_{x,y})$ respectively, we have that

$$\left(1 - \frac{\|v\|_A}{\|u\|_A}\cos(\theta_{u,v})\right)_{1A} < \left(1 - \frac{\|y\|_B}{\|x\|_B}\cos(\theta_{x,y})\right)_{1B} + \frac{\epsilon_2^2 - \epsilon_1^2}{2}$$

$$\Longrightarrow$$

$$\frac{\|v\|_A}{\|u\|_A}\cos(\theta_{u,v}) > \frac{\|y\|_B}{\|x\|_B}\cos(\theta_{x,y}) - \frac{\epsilon_2^2 - \epsilon_1^2}{2} \ .$$

Therefore, because the width of the gap size is

$$\epsilon_2^2 - \epsilon_1^2 > \frac{\epsilon_2^2 - \epsilon_1^2}{2} > 0 \ , \text{ and } \ \frac{\|y\|_B}{\|x\|_B} = \frac{\|v\|_A}{\|u\|_A} \ ,$$

we have that $\cos(\theta_{u,v})$ must be greater than $\cos(\theta_{x,y})$. Furthermore, the angle between two vectors can always be expressed as a value between 0 and π inclusive, and cosine decreases monotonically within that range. Thus if $\theta_{u,v}$ and $\theta_{x,y}$ are each expressed as an angle between 0 and π, we have that $\theta_{u,v} < \theta_{x,y}$. □

The use of the scaling factors δ_A and δ_B in the lemma with arbitrary vectors can introduce a significant amount of error. In the definition of gradient step-consistency the scaling factors serve as a normalization term for differing step sizes and are counterbalanced by their effect on the distance between the gradient vectors involved. Namely, as $\delta \to 0$, we have that $\|v\| \to \|u\|$. The lemma we just proved used arbitrary vectors without this requirement, introducing a scaling constant on an error term in inequality (2.3) that can be quite large when either δ is small.

inequality (2.4) doesn't contain the approximations used to bound the error for 2.3 and properly represents the relationship between the δ terms and the error that is introduced by them. In this case the scaling factors divide out cleanly if they are the same and no error is introduced. However, if the scaling factors are different, the error terms cannot approach 0 no matter how close the corresponding vector-norm ratios become. Because of this we will need the requirement in our theorem that the step sizes for each network are the same for any given inequality. This will not significantly weaken our results however.

Assuming the scaling factors are the same, this lemma gives us that any vectors $u, v \in (\mathbb{R}^m, \|\cdot\|_A)$ and $x, y \in (\mathbb{R}^n, \|\cdot\|_B)$ that satisfy inequality (2.2)

$$\frac{\|u - v\|_A}{\|u\|_A} < \frac{\|x - y\|_B}{\|x\|_B} \tag{2.2}$$

also satisfy the following inequalities:

$$\left(\frac{\|u\|_A^2 - \langle u,v\rangle_A}{\|u\|_A^2}\right) < \left(\frac{\|x\|_B^2 - \langle x,y\rangle_B}{\|x\|_B^2}\right) + \frac{1}{2}\left(\left(\frac{\|y\|^2}{\|x\|^2} - \frac{\|v\|^2}{\|u\|^2}\right) + (\epsilon_2 - \epsilon_1)\right) \tag{2.4}$$

$$= \left(\frac{\|x\|_B^2 - \langle x,y\rangle_B}{\|x\|_B^2}\right) + \left(\frac{\epsilon_{2^*}^2 - \epsilon_{1^*}^2}{2} + (\epsilon_{2^*} - \epsilon_{1^*}) + \frac{\epsilon_2 - \epsilon_1}{2}\right) \tag{2.10}$$

$$\text{where } \epsilon_1 = \frac{\|u-v\|_A}{\|u\|_A}\ , \qquad \epsilon_{1^*} = \frac{\|u\|_A - \|v\|_A}{\|u\|_A}\ ,$$

$$\epsilon_2 = \frac{\|x-y\|_B}{\|x\|_B}\ , \text{ and } \epsilon_{2^*} = \frac{\|x\|_B - \|y\|_B}{\|x\|_B}\ .$$

The lemma asserts that these error terms are $\mathcal{O}(\epsilon_2)$ or $\mathcal{O}(\epsilon_2^2)$, depending on whether ϵ_2 is greater than or less than 1. This is certainly true. However, distributions with certain qualities, such as when the angle $\theta_{u,v}$ is known to be small, can expect significantly less error. In particular, for any case where

$$\frac{\|y\|_B^2}{\|x\|_B^2} - \frac{\|v\|_A^2}{\|u\|_A^2} < \epsilon_2 - \epsilon_1 \text{ or equivalently } \epsilon_1 - \frac{\|v\|_A^2}{\|u\|_A^2} < \epsilon_2 - \frac{\|y\|_B^2}{\|x\|_B^2}\ ,$$

the lemma guarantees that inequality (2.3),

$$\frac{\|u\|_A^2 - \langle u,v\rangle_A}{\|u\|_A^2} < \frac{\|x\|_B^2 - \langle x,y\rangle_B}{\|y\|_B^2} + \epsilon + \epsilon^2\ , \tag{2.3}$$

will hold even with the assumption that $\epsilon = 0$. This will be used as an assumption in the following proof. While the additional assumption is necessary to rigorously prove the theorem, it is also likely to occur in any case where one network demonstrates higher step-consistency than another. Following the proof we will demonstrate empirically that this is the case in a variety of simulated circumstances (see figures 2.1 and 2.2).

Now that our lemma has been established and gradient step-consistency is more formally defined, we can move on to our primary theorem concerning gradient step-consistency.

Theorem 2.3 (Expected Performance in Step-Consistent Networks)**.** *Let two neural network architectures be given, one of which (network A) exhibits higher gradient step-consistency*

than the other (network B) for a given $\epsilon > 0$ within some context (including a shared dataset $\mathbb{X} = \mathbb{Y}$ or two datasets $\mathbb{X} \neq \mathbb{Y}$). Using the notation from Definition 2.1, if the networks satisfy the additional constraint that

$$\begin{aligned} & \mathop{\mathbb{E}}_{\phi\in\Phi,\chi\in B(\mathbb{X})}\left[\frac{\left\|\overline{h_{\mathbb{W}_A}(\phi,\chi)}-\overline{h_{\mathbb{W}_A}(\phi-\alpha,\chi)}\right\|_A}{\|\alpha\|_A}-\frac{\left\|\overline{h_{\mathbb{W}_A}(\phi-\alpha,\chi)}\right\|_A}{\|\alpha\|_A}\right] \\ < & \mathop{\mathbb{E}}_{\psi\in\Psi,\gamma\in B(\mathbb{Y})}\left[\frac{\left\|\overline{h_{\mathbb{W}_B}(\psi,\gamma)}-\overline{h_{\mathbb{W}_B}(\psi-\beta,\gamma)}\right\|_B}{\|\beta\|_B}-\frac{\left\|\overline{h_{\mathbb{W}_B}(\psi-\beta,\gamma)}\right\|_B}{\|\beta\|_B}\right] \\ & \qquad\qquad \textit{where}\ \ \alpha = \delta\ \overline{h_{\mathbb{W}_A}(\phi,\chi)} \\ & \qquad\qquad\qquad\ \ \beta = \delta\ \overline{h_{\mathbb{W}_B}(\psi,\gamma)}\ , \end{aligned} \tag{2.11}$$

for any shared step size ($\delta < \epsilon$) and regions ($\Phi \in (\mathbb{W}_A, \|\cdot\|_2)$ and $\Psi \in (\mathbb{W}_B, \|\cdot\|_2)$) that are contained within the provided context, then the following relationship also holds for those step sizes and regions within the context:

$$\begin{aligned} & \mathop{\mathbb{E}}_{\phi\in\Phi,\chi\in B(\mathbb{X})}\left[\frac{d\mathcal{L}_\phi(\overline{h_{\mathbb{W}_A}(\phi,\chi)})-d\mathcal{L}_{\phi-\alpha}(\overline{h_{\mathbb{W}_A}(\phi,\chi)})}{\delta d\mathcal{L}_\phi(\overline{h_{\mathbb{W}_A}(\phi,\chi)})}\right] \\ < & \mathop{\mathbb{E}}_{\psi\in\Psi,\gamma\in B(\mathbb{Y})}\left[\frac{d\mathcal{L}_\psi(\overline{h_{\mathbb{W}_B}(\psi,\gamma)})-d\mathcal{L}_{\psi-\beta}(\overline{h_{\mathbb{W}_B}(\psi,\gamma)})}{\delta d\mathcal{L}_\psi(\overline{h_{\mathbb{W}_B}(\psi,\gamma)})}\right] \end{aligned} \tag{2.12}$$

where $d\mathcal{L}_{w_i}(w_j)$ is the directional derivative of the loss function $\mathcal{L}$ at a point $w_1 \in \mathbb{W}$ taken in the direction of $w_j \in \mathbb{W}$.

In other words, the derivative of the loss function in the direction of a step taken decreases proportionally less between the two endpoints of the step for network A than for network B. This means that the expected decrease in loss for a step of any size less than ϵ is greater for network A than for network B. More formally stated:

$$\mathop{\mathbb{E}}_{\phi\in\Phi,\chi\in B(\mathbb{X})}\left[\frac{\overline{\mathcal{L}_A(\phi,\chi)}-\overline{\mathcal{L}_A(\phi-\alpha,\chi)}}{\delta\mathcal{L}_\phi(\overline{h_{\mathbb{W}_A}(\phi,\chi)})}\right] > \mathop{\mathbb{E}}_{\psi\in\Psi,\gamma\in B(\mathbb{Y})}\left[\frac{\overline{\mathcal{L}_B(\psi,\gamma)}-\overline{\mathcal{L}_B(\psi-\beta,\gamma)}}{\delta\mathcal{L}_\psi(\overline{h_{\mathbb{W}_B}(\psi,\gamma)})}\right]\ . \tag{2.13}$$

In short, greater gradient step-consistency increases the expected gain from a small step

during optimization, bringing it closer to the ideal result expected from a linearly projected step of the same size.

Proof. Assume the hypothesis. By the definition of the gradient, for any differentiable function $f : \mathbb{R}^n \to \mathbb{R}$, point $x \in \mathbb{R}^n$, and direction vector $v \in \mathbb{R}^n$, the gradient of f at x, $(\nabla f)_x$, is related to the directional derivative of f at x in the direction of v, $df_x(v)$, as follows:

$$\langle (\nabla f)_x, v \rangle_A = df_x(v) \ . \tag{2.14}$$

If we let $v = (\nabla f)_x$ that gives us

$$\|(\nabla f)_x\|^2 = \langle (\nabla f)_x, (\nabla f)_x \rangle_B = df_x((\nabla f)_x) \ , \tag{2.15}$$

and the directional derivative in the direction of the gradient is equal to the magnitude of the gradient squared.

Additionally, all of the conditions for equation (2.3) in Lemma 2.2 are satisfied by the theorem conditions. Furthermore, the second requirement of this theorem, equation (2.11), satisfies the additional condition in Lemma 2.2 that allows us to set all error terms in equation (2.3) to zero (the terms dependent on the step size are zero because the theorem statement requires that the same step size is used). This means that

$$\mathop{\mathbb{E}}_{\phi\in\Phi,\chi\in B(\mathbb{X})} \left[\frac{\left\| \overline{h_{\mathbb{W}_A}(\phi,\chi)} \right\|_A^2 - \left\langle \overline{h_{\mathbb{W}_A}(\phi,\chi)},\ \overline{h_{\mathbb{W}_A}(\phi-\alpha,\chi)} \right\rangle_A}{\|\alpha\|_A^2} \right] \tag{2.16}$$

$$< \mathop{\mathbb{E}}_{\psi\in\Psi,\gamma\in B(\mathbb{Y})} \left[\frac{\left\| \overline{h_{\mathbb{W}_B^2}(\psi,\gamma)} \right\|_B - \left\langle \overline{h_{\mathbb{W}_B}(\psi,\gamma)},\ \overline{h_{\mathbb{W}_B}(\psi-\beta,\gamma)} \right\rangle_B}{\|\beta\|_B^2} \right] \ . \tag{2.17}$$

Applying equations 2.14 and 2.15 results in the desired relation:

$$\mathop{\mathbb{E}}_{\phi\in\Phi,\chi\in B(\mathbb{X})}\left[\frac{d\mathcal{L}_{\phi}(\overline{h_{\mathbb{W}_A}(\phi,\chi)})-d\mathcal{L}_{\phi-\alpha}(\overline{h_{\mathbb{W}_A}(\phi,\chi)})}{\delta d\mathcal{L}_{\phi}(\overline{h_{\mathbb{W}_A}(\phi,\chi)})}\right]$$
$$<\mathop{\mathbb{E}}_{\psi\in\Psi,\gamma\in B(\mathbb{Y})}\left[\frac{d\mathcal{L}_{\psi}(\overline{h_{\mathbb{W}_B}(\psi,\gamma)})-d\mathcal{L}_{\psi-\beta}(\overline{h_{\mathbb{W}_B}(\psi,\gamma)})}{\delta d\mathcal{L}_{\psi}(\overline{h_{\mathbb{W}_B}(\psi,\gamma)})}\right].$$

Therefore the expected change in the slope of the loss surface for a sufficiently small step is smaller in network A than in network B. Furthermore, because the expected change in the slope of the loss is smaller for every step size smaller than epsilon, we have that the error in a linear projection estimating the total change in loss using the slope at the beginning of the step is smaller for network A than for network B (the total error in a linear projection is equivalent to the integral of the change in slope across the length of the step). In other words:

$$\mathop{\mathbb{E}}_{\phi\in\Phi,\chi\in B(\mathbb{X})}\left[\frac{\delta_1\mathcal{L}_{\phi}(\overline{h_{\mathbb{W}_A}(\phi,\chi)})-\left(\overline{\mathcal{L}_A(\phi,\chi)}-\overline{\mathcal{L}_A(\phi-\alpha,\chi)}\right)}{\delta\mathcal{L}_{\phi}(\overline{h_{\mathbb{W}_A}(\phi,\chi)})}\right]$$
$$<\mathop{\mathbb{E}}_{\psi\in\Psi,\gamma\in B(\mathbb{Y})}\left[\frac{\delta_2\mathcal{L}_{\psi}(\overline{h_{\mathbb{W}_B}(\psi,\gamma)})-\left(\overline{\mathcal{L}_B(\psi,\gamma}-\overline{\mathcal{L}_B(\psi-\beta,\gamma)}\right)}{\delta\mathcal{L}_{\psi}(\overline{h_{\mathbb{W}_B}(\psi,\gamma)})}\right]. \tag{2.18}$$

Our final desired result follows because a linear projection of the loss in the direction of the gradient for a small step of a given size is the most accurate projection that can be made with a first order derivative and it maximizes the expected change in the loss for a step of that size relative to the magnitude of the loss derivative:

$$\mathop{\mathbb{E}}_{\phi\in\Phi,\chi\in B(\mathbb{X})}\left[\frac{\overline{\mathcal{L}_A(\phi,\chi)}-\overline{\mathcal{L}_A(\phi-\alpha,\chi)}}{\delta\mathcal{L}_{\phi}(\overline{h_{\mathbb{W}_A}(\phi,\chi)})}\right] > \mathop{\mathbb{E}}_{\psi\in\Psi,\gamma\in B(\mathbb{Y})}\left[\frac{\overline{\mathcal{L}_B(\psi,\gamma)}-\overline{\mathcal{L}_B(\psi-\beta,\gamma)}}{\delta\mathcal{L}_{\psi}(\overline{h_{\mathbb{W}_B}(\psi,\gamma)})}\right]. \tag{2.13}$$

□

When gradient step-consistency is high the expected decrease in loss for a sufficiently small step in the direction of the negative gradient changes more slowly with increasing step size. In expectation this results in a greater decrease in loss for small steps of a fixed size because the direction of the negative gradient maximizes the decrease in loss in the limit as the step size approaches zero.

In practice, this means that for networks with high gradient step-consistency, larger step sizes can be used reliably and the expected decrease in loss for a step of fixed size is closer to the best expected gain from a step of that size given only first order derivative information.

When gradient step-consistency is low we can increase the stability of SGD by decreasing the step size. However, this has no effect on the underlying shape of the loss surface that causes poor gradient step-inconsistency, and when gradient step-consistency is too low SGD may fail to converge regardless of step size.

2.1.1 Requirements of Theorem 2.3. In theorem 2.3 two assumptions were made in addition to the primary requirement that the two networks involved had demonstrated a clear difference in expected step-consistency. First of all, it was assumed that the same step size is used in each case. Because the theorem results are true for any step size less than ϵ however, this did not weaken the results at all. However, the second assumption, that

$$\begin{aligned} & \mathop{\mathbb{E}}_{\phi\in\Phi,\chi\in B(\mathbb{X})}\left[\frac{\left\|\overline{h_{\mathbb{W}_A}(\phi,\chi)}-\overline{h_{\mathbb{W}_A}(\phi-\alpha,\chi)}\right\|_A}{\|\alpha\|_A}-\frac{\left\|\overline{h_{\mathbb{W}_A}(\phi-\alpha,\chi)}\right\|_A}{\|\alpha\|_A}\right] \\ < & \mathop{\mathbb{E}}_{\psi\in\Psi,\gamma\in B(\mathbb{Y})}\left[\frac{\left\|\overline{h_{\mathbb{W}_B}(\psi,\gamma)}-\overline{h_{\mathbb{W}_B}(\psi-\beta,\gamma)}\right\|_B}{\|\beta\|_B}-\frac{\left\|\overline{h_{\mathbb{W}_B}(\psi-\beta,\gamma)}\right\|_B}{\|\beta\|_B}\right] \end{aligned} \tag{2.11}$$

adds a potentially significant hurdle to applying the theorem. When the same step size is used this basically amounts to the statement

$$\mathop{\mathbb{E}}_{\phi\in\Phi,\chi\in B(\mathbb{X})}\left[\frac{\left\|\overline{h_{\mathbb{W}_A}(\phi-\alpha,\chi)}\right\|_A}{\left\|\overline{h_{\mathbb{W}_A}(\phi,\chi)}\right\|_A}\right]\approx\mathop{\mathbb{E}}_{\psi\in\Psi,\gamma\in B(\mathbb{Y})}\left[\frac{\left\|\overline{h_{\mathbb{W}_B}(\psi-\beta,\gamma)}\right\|_B}{\left\|\overline{h_{\mathbb{W}_B}(\psi,\gamma)}\right\|_B}\right].$$

As long as these two terms are sufficiently close they will not reverse the direction of the inequality. When the angle between each pair of vectors sampled from two given distributions is consistently small, we believe this should be true a significant portion of the time, which would make the second assumption unnecessary in many circumstances to expect the results of theorem 2.3.

In order to test this hypothesis we ran a simulation. Parameters for 1000 distributions were randomly generated and the distance between the two sides of inequality (2.2) and inequality (2.3) (ignoring the error terms) were measured. We will refer to inequality (2.2) as the condition inequality and inequality (2.3) as the result inequality.

The direction that the distance was measured for the result inequality was determined by the direction that the condition inequality held true (in expectation). In other words, the condition inequality distance is always positive and the second inequality distance is positive only if the direction the inequality is satisfied is the same direction as the condition inequality. We also measured the percentage of the time that the direction of each inequality for individual samples agreed with the direction of the inequality in expectation. The results can be seen in figure 2.1.

Notably, the direction of the condition and result inequality was the same in expectation for every single one of the 1000 generated distributions. Furthermore, there was an extremely strong correlation between the percent of the time that the inequality was satisfied for individual samples (in the same direction as the inequality in expectation) for the condition and result inequality. As might be expected, there were extremely few distributions that had lower than fifty percent sample-wise agreement with the expected inequality direction in both cases. However, in every single one of these cases, the direction of each expectation inequality still agreed.

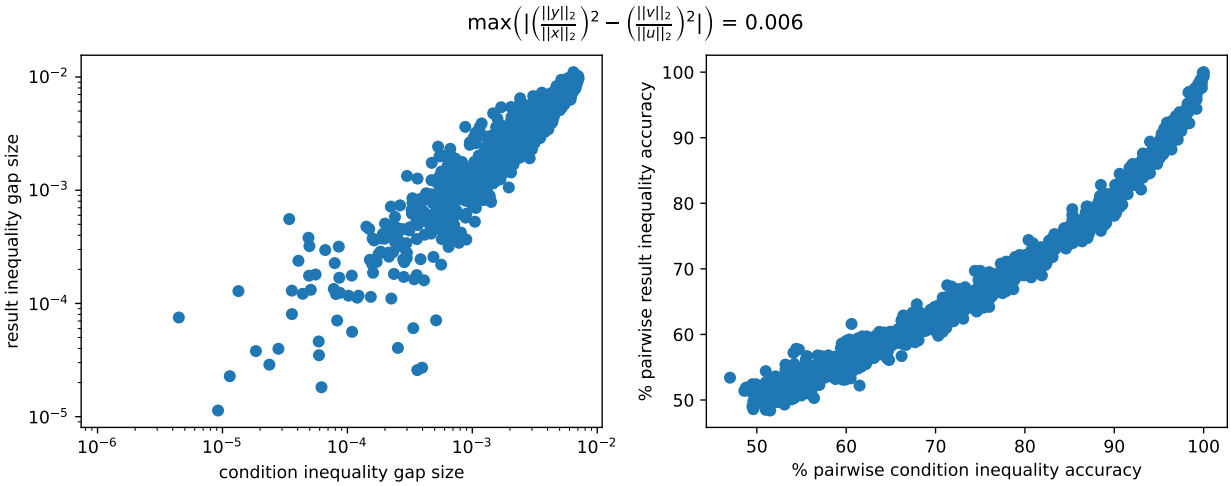


Figure 2.1: Simulation Results for Lemma 2.2. Using 1000 pairs of normal distributions with uniformly sampled parameters, we calculated the condition and result inequalities from Lemma 2.2 in expectation over 1000 sets of sampled vectors. Every single sampled distribution had the same direction for both inequalities in expectation and there was an extremely strong correlation between the percentage of the time that the direction of individual-sample inequalities matched the direction of the corresponding expectation inequality.

Vector samples for the simulation results displayed in figure 2.1 were generated by randomly sampling the first vector in each pair, u or x in the notation of Lemma 2.2, from a normal distribution and then perturbing it by another randomly sampled vector to obtain v or y. The scale and variance for each distribution used was sampled at the beginning of each of the 1000 sub-simulations. The scale of the perturbation vector for each distribution was sampled uniformly between zero and one-fifth of the scale of u or x respectively.

In order to more thoroughly strain the condition that

$$\frac{\|v\|_A}{\|u\|_A} \approx \frac{\|y\|_B}{\|x\|_B}$$

we ran the simulation a second time. During the second run we sampled a re-scaling factor for each of the four vectors between .5 and 1.5. By multiplying this by the distribution perturbation scale factor discussed previously (zero - one-fifth), we obtained a length bias which would allow $\frac{\|y\|_B}{\|x\|_B}$ and $\frac{\|v\|_A}{\|u\|_A}$ to vary by up to .4 without considering the compounding effect of perturbation size. Given that the width of the result inequality gap for every tested distribution inequality was less than this, that is a very large introduced bias. The results

can be seen in figure 2.2.

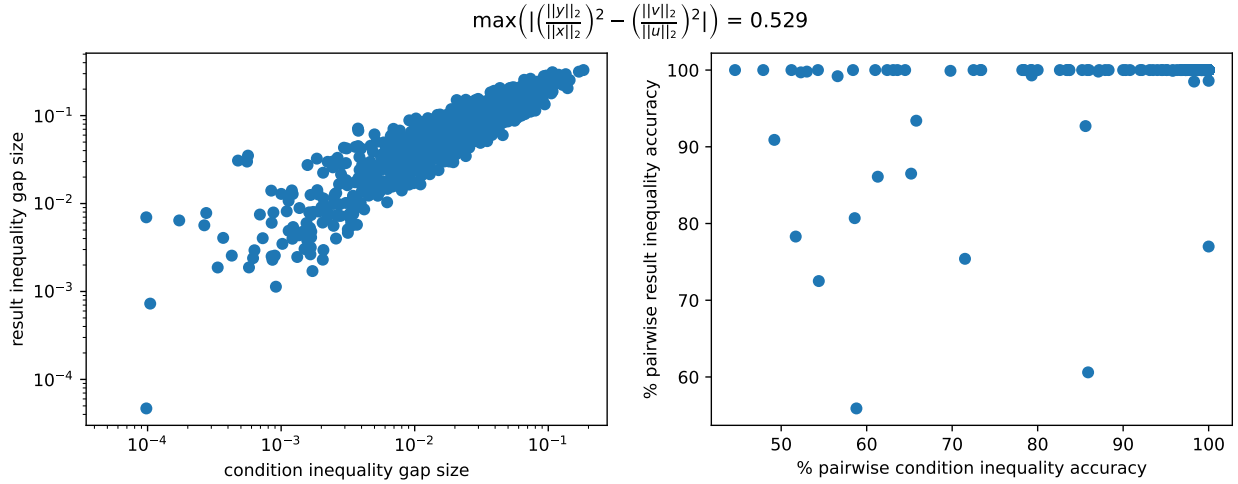


Figure 2.2: Magnitude Bias Simulation Results for Lemma 2.2. This simulation was run identically to that displayed in figure 2.1, with the additional inclusion that for each distribution a set stretch factors was calculated and applied to each vector sample. This resulted in a significant amount of introduced error coming from differing vector lengths. However, the condition and result inequalities from Lemma 2.2 were still in the same direction in expectation for every single one of the 1000 pairs of distributions tested.

Interestingly, adding a vector bias tended to increase the width of both inequality gaps significantly, which means that the error term

$$\frac{1}{2}\left(\frac{\|y\|^2}{\|\delta_B x\|^2} - \frac{\|v\|^2}{\|\delta_A u\|^2}\right)$$

in inequality (2.4)

$$\left(\frac{\|u\|_A^2 - \langle u, v\rangle_A}{\|\delta_A u\|_A^2}\right) < \left(\frac{\|x\|_B^2 - \langle x, y\rangle_B}{\|\delta_B x\|_B^2}\right) + \frac{1}{2}\left(\left(\frac{\|y\|^2}{\|\delta_B x\|^2} - \frac{\|v\|^2}{\|\delta_A u\|^2}\right) + (\epsilon_2 - \epsilon_1)\right) \quad (2.4)$$

$$\text{where} \quad \epsilon_1 = \frac{\|u - v\|_A}{\|\delta_A u\|_A} \quad \text{and} \quad \epsilon_2 = \frac{\|x - y\|_B}{\|\delta_B x\|_B}$$

was either small relative to resulting increase in the condition inequality gap, or reinforced the result inequality rather than undermining it, or both. There is a slight increase in this second test in the slope of the correlation line, which would indicate that the error terms reinforce the result inequality slightly in expectation. Furthermore, the percent of the time

the sample-wise result inequality matched the expectation inequality increased significantly.

This simulation was clearly not exhaustive with regards to vector sampling, and was not run with actual gradient samples from neural networks satisfying the step-consistency condition in theorem 2.3. However, the result set collected seems to indicate in the bounded perturbation case we care about that the additional requirement in theorem 2.3 we included to satisfy 2.2 is extremely likely to be true if the primary requirement of the theorem regarding step-consistency is true.

2.2 Gradient Conditioning with Respect to $\mathbb{W}$

Theorem 2.3 gave us some useful and intuitive insights into the expected training performance of networks that have high gradient step-consistency. However, the proof of each of these results relied heavily upon the expectation found in equation (2.1). In other words, while theorem 2.3 contains powerful statements about the average performance of an architecture relative to another within the provided context, it provides very little information about the consistency with which these performance improvements can be expected and allows for a large amount of variance in performance. Naturally, because SGD is an iterative process, this variance matters quite a bit.

Consider equation (2.1) below:

$$\mathop{\mathbb{E}}_{\phi\in\Phi,\chi\in B(\mathbb{X})}\left[\frac{\left\|\overline{h_{\mathbb{W}_A}(\phi,\chi)}-\overline{h_{\mathbb{W}_A}(\phi-\alpha,\chi)}\right\|_2}{\left\|\overline{h_{\mathbb{W}_A}(\phi,\chi)}\right\|_2}\right] \tag{2.1}$$

$$< \mathop{\mathbb{E}}_{\psi\in\Psi,\gamma\in B(\mathbb{Y})}\left[\frac{\left\|\overline{h_{\mathbb{W}_B}(\psi,\gamma)}-\overline{h_{\mathbb{W}_B}(\psi-\beta,\gamma)}\right\|_2}{\left\|\overline{h_{\mathbb{W}_B}(\psi,\gamma)_2}\right\|}\right] \tag{2.19}$$

$$\text{where}\quad \alpha \;=\; \delta_1\;\overline{h_{\mathbb{W}_A}(\phi,\chi)}$$

$$\text{and}\quad \beta \;=\; \delta_2\overline{h_{\mathbb{W}_B}(\psi,\gamma)}\;.$$

In order to define a metric on gradient step-consistency that can be used more reliably

for networks in any context, we need to place a restriction on the size of step that is allowed so that we achieve more repeatable results. Unfortunately, the radius of the region in which linear projections of the loss are reliable could vary quite a bit for different networks. Furthermore, because the step α is defined relative to the magnitude of the gradient, using the learning rate δ_1 is not sufficient to guarantee that step size for a given learning rate is comparable between the weight spaces of two different networks, nor would it be desireable to need to use the same learning rate in each case. We can fix the second issue very easily by adding the normalization term to the quantity within each expectation:

$$\frac{\left\|\overline{h_{\mathbb{W}_A}(\phi,\chi)} - \overline{h_{\mathbb{W}_A}(\phi-\alpha,\chi)}\right\|_2}{\left\|\overline{h_{\mathbb{W}_A}(\phi,\chi)}\right\|_2} \cdot \frac{\|\phi\|_2}{\|\alpha\|_2} \,.$$

Creating a metric using this term that accounts for variance caused by differences in direction is more difficult. If we were estimating change in the loss caused by a step of fixed size stepping in the direction of the gradient would always result in highest possible change in the loss. Therefore, this quantity would very effectively become a representation of the highest possible change induced by a step of a given size at that weight. However, just because the gradient direction maximizes the change in loss or a small step does not mean that it maximizes the change in the gradient for a step of that size. In other words, the radius for which a gradient is expected to maximize the change in loss for a step inside of that radius might be smaller for some networks than for others. In other words, we will need to use a limit on the step size to ensure that our quantity is always measured inside of this meaningful radius.

Fortunately, and perhaps unsurprisingly given the amount of attention given condition numbers in chapter 1, our normalization of the gradient step-consistency quantity in Definition 2.1 means it is now equivalent to the quantity measured in the relative condition number of the gradient function when restricted to changes in the weight space. The limit and supremum found in the relative condition number mean that we can measure the quantity we are

interested in across a wide variety of networks and datasets and expect consistency in the result. We define the relative condition number in this context below:

Definition 2.4 (**Relative Condition Number with Respect to $\mathbb{W}$**)**.** Let an architecture $g : ((\mathbb{R}^m, ||\cdot||_{\mathbb{W}}), (\mathbb{R}^k, ||\cdot||_{\mathbb{X}})) \to \mathbb{R}^j$ and a loss function $f : (\mathbb{R}^j, T^h) \to \mathbb{R}$ be given. Using the notation of Definition 1.3, the relative gradient condition number $\kappa_{h_{\mathbb{W}},\mathbb{W}}(w)$, $w \in \mathbb{W}$, of $h_{\mathbb{W}}$ with respect to the weight space $\mathbb{W}$ is defined as follows:

$$\kappa_{h_{\mathbb{W}},\mathbb{W}}(w) = \hat{\kappa}_{h_{\mathbb{W}},\mathbb{W}}(w) \cdot \frac{||w||_{\mathbb{W}}}{||h_{\mathbb{W}}(w)||_{\mathbb{W}}} = \frac{||H_{f,\mathbb{W}}(w)||_{\mathbb{W},\mathbb{W}} \cdot ||w||_{\mathbb{W}}}{||h_{\mathbb{W}}(w)||_{\mathbb{W}}} \tag{2.20}$$

$$\text{where} \tag{2.21}$$

$$\hat{\kappa}_{h_{\mathbb{W}},\mathbb{W}}(w) = \lim_{\epsilon \to 0} \sup_{||\omega||_{\mathbb{W}} < \epsilon} \frac{||h_{\mathbb{W}}(w+\omega) - h_{\mathbb{W}}(w)||_{\mathbb{W}}}{||\omega||_{\mathbb{W}}} = ||J_{h_{\mathbb{W}},\mathbb{W}}(w)||_{\mathbb{W},\mathbb{W}} = ||H_{f,\mathbb{W}}(w)||_{\mathbb{W},\mathbb{W}} \ , \tag{2.22}$$

$J_{h_{\mathbb{W}},\mathbb{W}}(w)$ is the Jacobian of $h_{\mathbb{W}}$ with respect to $\mathbb{W}$ at w, $H_{f,\mathbb{W}}(w)$ is the Hessian of the loss $(f \circ g)$ with respect to $\mathbb{W}$ at w, and $||\cdot||_{\mathbb{W},\mathbb{W}}$ is the induced norm on transformations between the weight space and itself. The second equality in each case holds if and only if $J_{h_{\mathbb{W}},\mathbb{W}}(w)$ is defined at w.

$\kappa_{h_{\mathbb{W}},\mathbb{W}}(w)$ is a measure of the worst-case step-inconsistency of individual gradient estimates during learning for a given point in weight space. The limit and supremum make it difficult to estimate numerically however. However, we have a theorem in chapter 1 that makes it possible to estimate it relatively easily.

By theorem 1.6, if $h_{\mathbb{W}}(w)$ is differentiable with respect to $\mathbb{W}$ at w, then an $\epsilon > 0$ exists such that

$$\kappa_{h_{\mathbb{W}},\mathbb{W}}(w) + o(\epsilon) \geq \frac{||h_{\mathbb{W}}(w+\omega) - h_{\mathbb{W}}(w)||_{\mathbb{W}}||w||_{\mathbb{W}}}{||\omega||_{\mathbb{W}}||h_{\mathbb{W}}(w)||_{\mathbb{W}}} \tag{2.23}$$

for any $\omega \in \mathbb{W}$ where $||\omega||_{\mathbb{W}} < \epsilon$.

This allows us to define a viable metric for gradient step-consistency that does not rely on an expectation, a limit, or a supremum: the approximate best lower bound on $\kappa_{h_{\mathbb{W}},\mathbb{W}}$:

Definition 2.5 (Approximate Best Lower Bound on $\kappa_{h_{\mathbb{W}},\mathbb{W}}$)**.** Let $\epsilon > 0$ and $n \in \mathbb{N}$ be given and let $B_\epsilon(0) = \{p \in \mathbb{W} : 0 < ||p||_{\mathbb{W}} < \epsilon\}$. Then we define our approximate best lower bound on $\kappa_{h_{\mathbb{W}},\mathbb{W}}$ as

$$\kappa^{\gtrapprox}_{h_{\mathbb{W}},\mathbb{W}}(w) := \max_{\omega_k \in B_\epsilon(0), 0 \leq k < n} \left(\frac{||\overline{h_{\mathbb{W}}(w+\omega_k, \chi_j)} - \overline{h_{\mathbb{W}}(w, \chi_i)}||_{\mathbb{W}} ||w||_{\mathbb{W}}}{||\omega_k||_{\mathbb{W}} ||\overline{h_{\mathbb{W}}(w, \chi_i)}||_{\mathbb{W}}}, \ \chi_i, \chi_j \in B_b \right) . \quad (2.24)$$

Unless $i \neq j$ is clearly specified, this metric will always be used in this thesis with $i = j$. However, it is possible to establish a probabilistic bound on the error introduced by allowing $i \neq j$ using the law of large numbers with sufficiently large b.

The above metric is dependent on the batch-size b in two distinctly different ways. As b grows we have by the law of large numbers that $\overline{h_{\mathbb{W}}(w, \chi)}$ becomes increasingly likely to be close to $h_{\mathbb{W}}(w)$. In this regard, $\kappa^{\gtrapprox}_{h_{\mathbb{W}},\mathbb{W}}(w)$ with $i = j$, becomes a measure of the conditioning of the theoretical gradient $h_{\mathbb{W}}(w)$ as $b \to \infty$. However, the metric is equally valid for any b if viewed as a measure on the conditioning of $\overline{h_{\mathbb{W}}(w, \chi)}$, which is in fact the gradient estimation which is actually experienced during training.

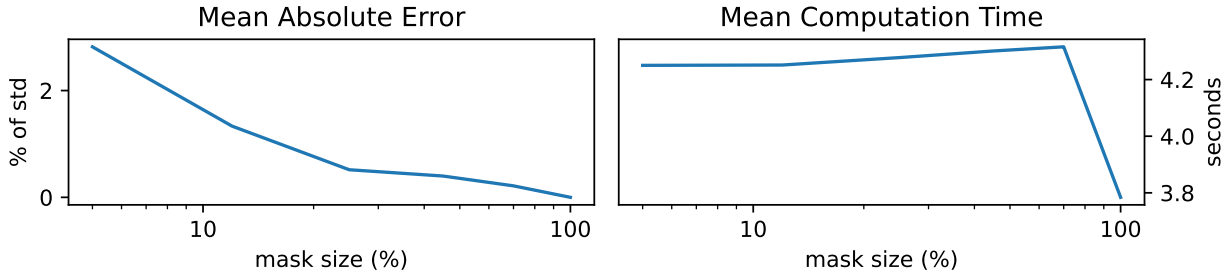


Figure 2.3: Mask Effectiveness for Gradient Conditioning w.r.t. $\mathbb{W}$

The second way that $\kappa^{\gtrapprox}_{h_{\mathbb{W}},\mathbb{W}}(w)$ is dependent on the batch-size b has already been stated. Namely, it is the confidence with which different batches, $i \neq j$, can be used without exceeding a specified amount of error. In practice this error can trivially be avoided simply by using the same batch $i = j$. However, we have allowed for the introduction of additional error into the metric in our definition for one very specific reason. When using a sample size $n = 1$, we can estimate $\kappa^{\gtrapprox}_{h_{\mathbb{W}},\mathbb{W}}(w)$ during training with very little additional cost as follows:

$$\kappa^{\gtrapprox}_{h_{\mathbb{W}},\mathbb{W}}(w) = \frac{||\overline{h_{\mathbb{W}}(w+\omega,\chi_j)} - \overline{h_{\mathbb{W}}(w,\chi_i)}||_{\mathbb{W}}||w||_{\mathbb{W}}}{lr||\overline{h_{\mathbb{W}}(w,\chi_i)}||^2_{\mathbb{W}}},\ \chi_i,\chi_j \in B_b$$

$$\text{where}\ \ \omega = \ -lr * \overline{h_{\mathbb{W}}(w,\chi_i)},\ lr > 0,\ lr \in \mathbb{R}\ .$$

It should be noted that estimating $\kappa^{\gtrapprox}_{h_{\mathbb{W}},\mathbb{W}}(w)$ with $i \neq j$ is not a valid way to measure the conditioning of $\overline{h_{\mathbb{W}}(w+\omega,\chi)}$ with respect to changes in the weights and inputs simultaneously because there is no modification to the normalizing terms in the condition number based on the magnitude of change in the input batch. While it would be possible to measure the conditioning of $h_{\mathbb{W}}(w,x,t)$ with respect to both $\mathbb{W}$ and $\mathbb{X}$ using a normalizing term of the form $||\text{concat}(w,x)||_\alpha$, the discrete nature of the subspace in $\mathbb{X}$ that can be paired with known labels in $\mathbb{T}^h$ (i.e. the training set) makes it impossible to reliably use the linear approximation applied in theorem 1.6 to estimate the conditioning with respect to $\mathbb{X}$ by using pairs of inputs or batches from the training set.

2.3 Generating Targeted Weight Perturbation Samples

A variety of methods could be employed to generate samples of $\omega_k \in B_\epsilon(0)$ in equation (2.5). Randomly sampling directions is easy and often effective. Furthermore, when drawing a limited number of samples in a space as high dimensional as the weight space, each sample is nearly guaranteed to be orthogonal to the others. However, in an over-parameterized network, there are several reasons to believe that perturbing the weights in randomly sampled directions may have very little impact on the loss or gradient.

In networks that use rectifier activation functions, neurons have distinct active and inactive states. During a typical forward pass for a given input, a large number of neurons will not activate. Several studies have demonstrated that in over-parameterized classification networks small groups of neurons specialize to recognize particular features that are unique to a class or group of classes (for example: [20]). A small random weight perturbation is

unlikely to make groups of neurons that are unrelated to a given input activate. However, a targeted shift in weights for neurons that would already be active for that input otherwise, or to groups that are close to activating on that input, would have a much larger impact relative to the magnitude of the perturbation.

The insensitivity of the loss function to random perturbations in the weight space could be further complicated in classification problems by the small number of class labels and the limited number of samples within the evaluating batch. Within each batch the loss is evaluated over a relatively small number of samples. Any weight perturbation that does not significantly change the output for at least one sample in the evaluating batch will not change the loss or gradient very much. Furthermore, when the number of potential labels is small a large portion of the network output space results in the same classification and, depending on the loss function, a very similar loss value.

This potential insensitivity to random perturbations is not particularly relevant to the theoretical considerations of this paper. It does, however, mean that random sampling will likely not be sufficient to consistently get a tight lower bound on the maximum in equation (2.5). In order to avoid this issue, we have developed and tested a targeted method for generating weight perturbation samples $\omega_k \in B_\epsilon(0)$. This technique includes preferential treatment of the gradient direction.

Specifically, we will generate samples by estimating the gradient with a single input (or rather a batch of inputs at a time without averaging) as outlined in the following definition:

Definition 2.6 (**Targeted Samples Technique for Weight Perturbation Directions**)**.** Let a batch $\chi \in B_b(\mathbb{X})$ and a sample size $n < b$ be given. When generating weight perturbation direction samples, ω_i, where it is desireable to maximize the change in loss or in the gradient (such as ω in equation (2.5)), the samples will be drawn from the set

$$\left\{ \frac{\eta\ \zeta\ h_{\mathbb{W}}(w, x_i, t_i)}{\|h_{\mathbb{W}}(w, x_i, t_i)|)\|_{\mathbb{W}}} \mid \zeta \sim \mathcal{U}(\{-1, 1\}),\ h_{\mathbb{W}}(w, x_i, t_i) \in h_{\mathbb{W}}(w, \chi) \right\} \tag{2.25}$$

where $\eta \in \mathbb{R}$ is the desired scale. Unless otherwise specified we will use $\eta = .001$.

It should be noted that using samples generated in this way is highly memory intensive if the samples must be stored while other forward or backward passes are executed, as when calculating equation (2.5).

Perturbing in the direction of a weight-gradient calculated with a single input (or the batch-averaged gradient if individual gradients are not available) should have a significant impact on any elements of the evaluating batch from the same class. Furthermore, because we use a subset of the evaluating batch to generate these samples it will maximize the change in loss for those elements. Up to b samples can be drawn with minimal additional temporal cost because they can be computed in the backward pass for $|\overline{h_{\mathbb{W}}(w,\chi)}|$ which is already required to calculate $\kappa^{\gtrapprox}_{h_{\mathbb{W}},\mathbb{W}}(w)$.

This sampling technique should increase the rate at which the lower bound on the maximum in equation (2.5) is expected to initially converge over fully random samples and significantly reduce the number of samples required. Random sampling would be guaranteed to match or exceed the best lower bound identified using any bounded sampling method as $n \to \infty$, but our use case is necessarily restricted to a small number of samples and the initial convergence rate is much more important.

When the memory requirements of storing individual gradients for this sampling technique is too high, the following sampling mechanism may be used:

Definition 2.7 (**Limited Targeted Samples for Weight Perturbation Directions**)**.** Let a batch $\chi \in B_b(\mathbb{X})$ and a sample size $n < b$ be given. When generating weight perturbation direction samples, ω_i, where it is desireable to maximize the change in loss or in the gradient (such as ω in equation (2.5)), up to two samples will be drawn from the set

$$\left\{ \frac{\eta\ \overline{h_{\mathbb{W}}(w,\chi)}}{\left\|\overline{h_{\mathbb{W}}(w,\chi)}\right\|_\infty},\ \frac{-\eta\ \overline{h_{\mathbb{W}}(w,\chi)}}{\left\|\overline{h_{\mathbb{W}}(w,\chi)}\right\|_\infty} \right\} \tag{2.26}$$

where $\eta \in \mathbb{R}$ is the desired scale. Unless otherwise specified we will use $\eta = .001$.

We performed a test comparing these targeted sampling techniques to random sampling.

Elements were drawn from a normal distribution with the same mean and standard deviation as the weights being perturbed, with a scaling factor of $\eta = .001$. The results can be see in figure 2.4. Because using the batch-averaged gradient sample performed so well compared to using individual gradient samples and is so much more efficient, we opted to use that method for each of our other tests. This will also enable us to more easily compare results between small networks that are able to use individual gradient sampling without running out of available memory on the GPU with larger models that must use the batch-averaged gradient technique.

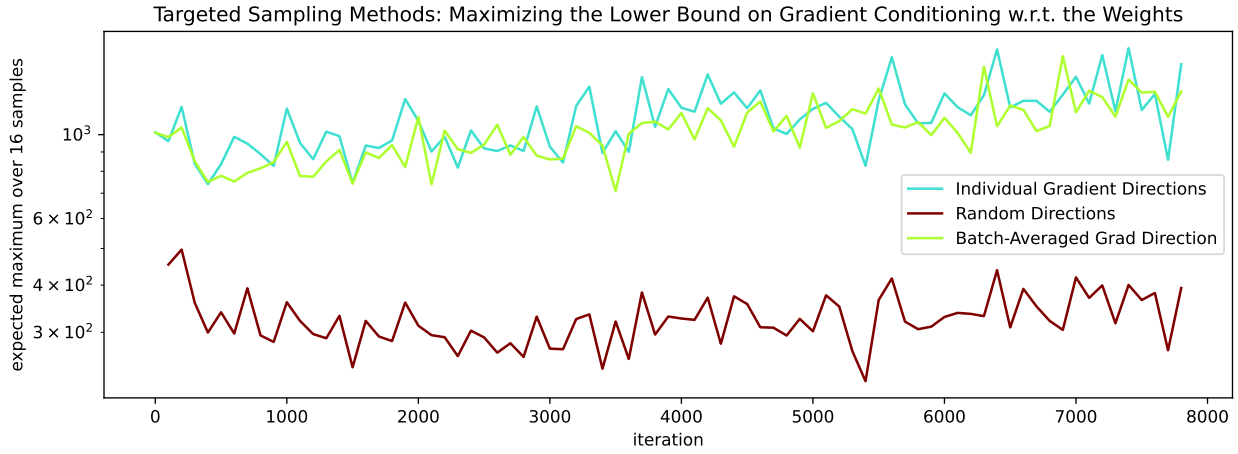


Figure 2.4: Targeted vs Random Sampling in $\mathbb{W}$. During the training of an 11 layer convolutional network (with normalization but without skip connections), 64 input perturbation directions were generated via the sampling methods described in section 2.3. Bootstrapping was used to estimate an expected maximum over a sample size of 16.

Using the batch-averaged gradient sampling technique, we measured the gradient conditioning with respect to the weight space for several networks throughout a full training run. The results can be seen in Figure 2.5.

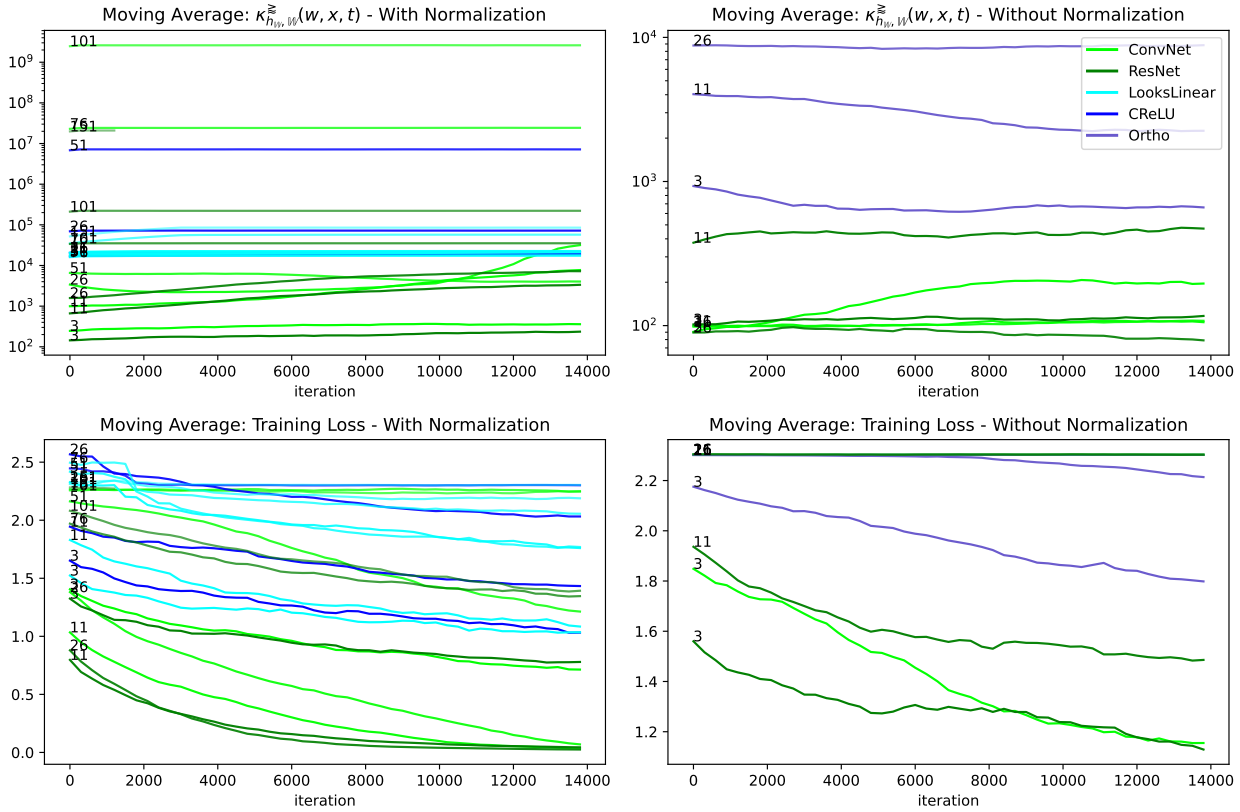


Figure 2.5: The Gradient Conditioning w.r.t. the Weight Space (see Definition 2.5) was measured throughout training for several different types of architectures. It tends to change slowly for networks that converge, but is consistent when no convergence occurs.
The architectures tested included fully connected networks with orthogonal initialization and the ReLU or CReLU activation function, the Looks-Linear architecture (see Definition 3.9), and convolutional networks with and without residual connections. The label on each line is the number of layers in the architecture.

For many of the networks we trained the gradient conditioning with respect to the weight space was relatively constant throughout training, although some changed rapidly at certain points.

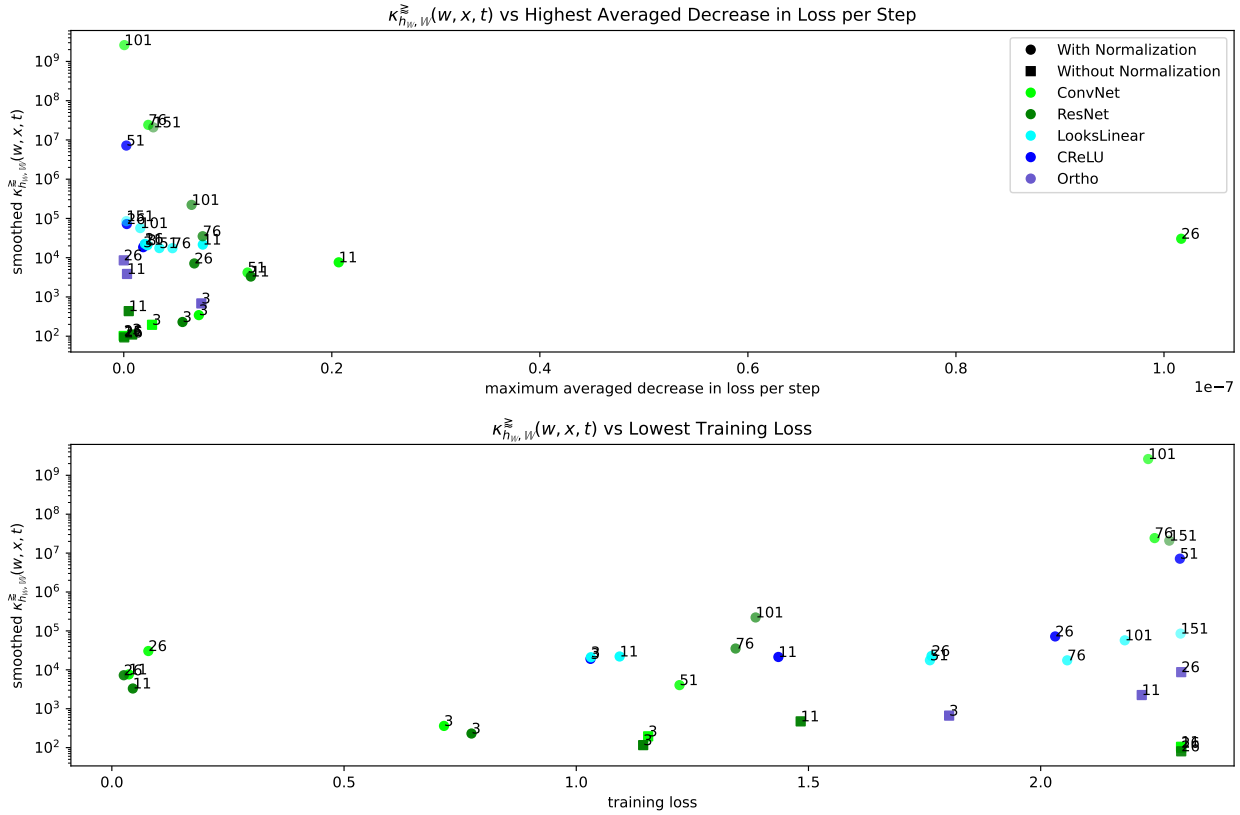


Figure 2.6: The Gradient Conditioning w.r.t. the Weight Space was measured at the point of steepest descent as well as at the point with the lowest loss (smoothed over a few hundred iterations in each case) and plotted as a point for each network. It can be seen clearly in this figure that networks with excessively high gradient conditioning w.r.t. the weight space fail to converge. The architectures tested include fully connected networks with orthogonal initialization and the ReLU or CReLU activation function, the Looks-Linear architecture (see Definition 3.9), and convolutional networks with and without residual connections. The label on each point is the number of layers in the architecture.

While the correlation between gradient conditioning in the weight space and the highest average decrease in loss (a moving average over 20 samples) is not as pronounced as we would like, it does seem clear that exceptionally poor conditioning results in a near complete lack of convergence.

2.4 Literature Related to Gradient Step-Consistency

There is a significant body of literature that is related to gradient step-consistency, generally belonging to one of two categories. The first category includes papers that study the smooth-

ness of the loss surface in limited regions. The usage of the term "smoothness" is colloquial in this case, although some papers have been able to establish more formal results. Because the parameter space of deep neural networks is so large, calculating the loss, evaluating smoothness, or generating visualizations are all extremely expensive for even small regions of weight space. Because of this, the region chosen for study is almost always restricted to the area immediately surrounding a local minimizer, which limits the use of these results and techniques for our purpose of evaluating learning effectiveness throughout training.

Furthermore, to reduce expense, the dimension of the loss surface is typically reduced by projecting down onto a basis of random vectors in the weight space prior to evaluating the loss over the target region. It was observed by Li et al., however, that fully-random directions are unlikely to observe sharp features in the loss surface [12]. Our own results when testing gradient conditioning in randomly generated weight directions in section 2.3 supports this hypothesis. Li et al., utilized a targeted method to construct directions for the basis of their loss surface visualization projections. However, as in our own tests, it is impossible to determine by the same method what features are not being observed by a projection without finding another targeted sample direction generation technique that is even more effective or exhaustively exploring the space. The second group of papers we are about to discuss may provide a way to evaluate the effectiveness of random projections at capturing sharp features more effectively.

All of these issues make exploration of the loss surface through lower-dimensional projections an unlikely candidate for evaluating the loss-smoothness or gradient-step consistency of a potential architecture.

The second body of work that is strongly related to gradient step-consistency includes papers studying the spectrum of the Hessian of the loss function with respect to the weight space. The relative condition number of the gradient with respect to the weight space, equation (2.4), is the norm of the Hessian with respect to the weight space multiplied by a normalizing constant. If the two-norm is used then this is the largest singular value of the

Hessian multiplied by a normalizing factor.

Calculating even the first few eigenvalues of the Hessian for large networks was prohibitively expensive until Ghorbani et al. published their algorithm for estimating the entire Hessian spectrum with high precision in 2019 [5]. This publication represented a distinct turning-point in studies of this nature. In the words of Ghorbani et al.:

> Previous studies on the Hessian have focused on small models, or are limited to computing only a few eigenvalues (Sagun et al., 2016; 2017; Yao et al., 2018). In the absence of such concrete information about the eigenvalue spectrum, many researchers have developed clever ad hoc methods to understand notions of smoothness, curvature, sharpness, and poor conditioning in the landscape of the loss surface. Examples of such work, where some surrogate is defined for the curvature, include the debate on flat vs sharp minima (Keskar et al., 2016; Dinh et al., 2017; Wu et al., 2017; Jastrzębski et al., 2017), explanations of the efficacy of residual connections (Li et al., 2018b [12]; Orhan & Pitkow, 2017 [14]) and batch normalization (Santurkar et al., 2018 [19]), the construction of low-energy paths between different local minima (Draxler et al., 2018), qualitative studies and visualizations of the loss surface (Goodfellow et al., 2014), and characterization of the intrinsic dimensionality of the loss (Li et al., 2018a; Fort & Scherlis, 2018). In each of these cases, detailed knowledge of the entire Hessian spectrum would surely be informative, if not decisive, in explaining the phenomena at hand.

Since the publication of Ghorbani et al.'s algorithm for estimating the full spectrum of the Hessian, a few papers have already been published containing more concrete results on the effect batch normalization, skip-connections, and other architectural traits have on the loss surface than were ever available previously. This method demonstrates extreme promise for learning how different architectural traits impact gradient stability and usefulness during training. However, it is well outside the scope of this thesis in terms of implementation

alignment with the other metrics and techniques used. Because of this we will restrict ourselves to a brief summary of related results and attempt to observe these findings in our own results.

Because the two methods common in the field are computationally expensive relative to the other methods we present, we will not be able to use them in our own tests. However, we will present a brief summary of relevant findings and identify the results in our own tests that would validate them. Our primary interest is in architectural traits that have been observed to have a smoothing effect on the loss surface, these being likely candidates for improving gradient step-consistency.

2.4.1 The Impact of Batch Normalization. In 2015 batch normalization was first introduced by Ioffe et al. [8]. They proposed it as a solution to an issue they named *internal covariate shift*, which is when the distribution of each output for a given layer changes throughout training. They argued that this caused issues for subsequent layers, which therefore had to continually adjust to a changing input. Their proposed solution, batch normalization, fixed this perceived issue by re-centering and re-scaling each output of a layer to fit a standard normal distribution across each batch.

Because inputs to a batch normalization (batch-norm) layer are each re-centered and re-scaled across an entire batch it doesn't neatly fit inside our definition of an architectural trait. It is influenced slightly during training by the batch-size, which is an optimization parameter. However, it does learn the bias and scaling corrections over time, meaning that a network containing batch normalization layers can be evaluated consistently on individual inputs. Several other normalization layers have been proposed since then that behave very similarly to batch-norm, including layer-norm, instance-norm, and group-norm [26]. Each of these others normalizes inputs across a portion or all of the layer instead of across the batch. It has been demonstrated that these can have superior performance to batch-norm when a small batch is used.

Batch normalization is extremely effective at improving training in many types of ar-

chitectures, and was widely adopted very quickly. It can significantly reduce training time and makes networks less sensitive to initialization. The reason it does these things has been disputed however. In 2015 Santurkar et al. argued that reducing internal covariate shift had very little to do with batch normalization's positive benefits. They demonstrated this by explicitly injecting covariate-shifted random noise after batch-norm layers and showing that the positive benefits of batch-norm persisted even with this injected noise [19]. They proposed instead that batch normalization smooths the loss surface, and therefore stabilizes gradients, allowing for faster training.

Ghorbani et al. in 2019 disputed this result using eigenvalues of the Hessian in networks with and without batch normalization [5]. Santurkar et al. clearly demonstrated that on a per-layer basis, the Lipschitz bound on change in the loss (which is similar to the absolute condition number of the loss but is over pairs of inputs without a limit on the distance between them) is reduced when batch normalization is used. However, Ghorbani et al. argue that this per-layer analysis impacts only a block-diagonal portion of the Hessian and is insufficient to establish smoother training generally.

As an example they train two identical networks with batch normalization. One by using the full-dataset to evaluate each gradient and the other using typically-sized batches. Each network has nearly identical largest eigenvalues and the magnitudes of the learned parameters in each case is similar. Because of this the two networks would have roughly equivalent gradient step-consistency and conditioning with respect to the weight space (see Definition 2.4). However, the network trained with gradients calculated using the full dataset learns much more slowly than the network trained with batches and the Hessian had a larger number of outlier eigenvalues. They conclude with the observation that the largest eigenvalue (and thus gradient step-consistency, assuming similarly sized parameters), is not sufficient to explain ease of training. Notably, the only difference between these two networks is the batch-size, which falls under our discussion of gradient batch-dissonance in chapter 3. The arguments made by Ghorbani et al. align well with our own argument that gradient

step-consistency and batch-dissonance are both critical to ease and speed of training.

Another paper from 2019 by Yao et al. [28] observed a similar suppression of outlier eigenvalues in deep networks, but demonstrated that this smoothing effect was largely absent in shallower networks. However, batch normalization demonstrated clear benefits in the shallow networks even without the suppression of outlier eigenvalues noted by Ghorbani et al. [5]. Wu et al. [25] and Papyan [15] confirm and explain this suppression of outlier eigenvalues more extensively.

2.4.2 The Impact of Skip-Connections. The second architectural trait that has received significant attention from these related papers is the skip connection. More specifically, the residual skip connection. A skip connection in a deep neural network is a method of transferring the output of a layer to a subsequent layer or layers without it passing through the layers in between. There are two common types types of skip connections: residual connections and concatenation connections.

Concatenation connections are the type of connection most directly implied by the name "skip connection". Effectively, the output of a chosen layer is provided without modification as input to another layer deeper in the network. This later layer then takes input from two layers, the layer immediately before it in the forward pass and the layer connected to it through the skip connection. In other words, the output from the layer connected by the skip connection is concatenated to the output of the previous layer. Since the development of residual connections however, these have featured much less prominently in related literature.

The other type, residual connections, are additive in nature. Instead of transforming the input, a residual connection layer or block operates on the input by passing it through one or more traditional layers, but instead of giving its own output to the next layer, it re-scales and it adds it to the original input. In a ResNet, or Residual Network, there are typically very few traditional layers that directly transform the input and they usually reside at the beginning and/or end of the network. Instead a single tensor is gradually transformed through repeated addition by residual blocks through the majority of the network.

Introduced in 2015 by He et al., residual connections are particularly effective at making the backward pass shallower by reducing the number of repeated applications of the chain rule. This is well known for making the backward pass more resilient against common gradient issues, such as vanishing or exploding gradient magnitudes.

There has been as much or more research about the exact mechanism behind the dramatic improvement in learning that can be caused by adding residual connections to an architecture as there has been about batch normalization.

In 2017 Li et al. [12] proposed using their surface-visualization technique that skip connections could dramatically smooth the loss surface. They demonstrated this for both types of skip connections we have mentioned, showing that ResNet architectures and an architecture called DenseNet [7] (which has a very large number of concatenation skip-connections) appear incredibly smooth in their projections compared to networks with a similar structure but without any skip connections. Naturally, both types of networks also compared very favorably in speed of training and final performance against the similar architectures without skip connections.

Another paper in 2017, by Ornhan et al. [14], demonstrated that residual connections reduce the prevalence of various types of singularities that slow learning. Yet another paper in 2017, this one by Masato Taki [22], demonstrated that ResNets are more robust against poor initialization than other similar architectures, especially when paired with batch normalization.

In 2018 Zaeemzadeh et al. demonstrated that residual connections are norm-preserving [30]. In other words, they demonstrate that adding residual connections causes the magnitude of the gradient to be more consistent throughout the backward pass, making the the gradient much more stable throughout learning.

The results from all of these papers that were published quickly after the introduction of residual connections indicated that skip connections dramatically smooth the loss surface, especially when paired with batch-normalization. However, in 2019 two Hessian-spectrum

papers, by Ghorbani and Yao [5, 28] presented a more nuanced picture. Both observed that adding residual connections in deep networks resulted in a rapid compression of all eigenvalues towards zero, resulting in a smoother loss surface and better conditioning. Furthermore, this effect is more dramatic in deeper networks, with the increase in eigenvalue magnitudes caused by removing residual connections being much more rapid in deeper networks. However, Yao and Ghorbani both observe that the smoothing effect of adding residual connections is reversed in shallow networks. This reversal was observed in networks smaller than 20 layers by Yao and in a 32-layer ResNet by Ghorbani. Moving forward we will make note of whether our own test results reflect these observations on skip-connections and batch-normalization in the literature.

Chapter 3. Gradient Batch-Dissonance

Chapter 2 reviewed gradient step-consistency and introduced theorems and related literature. This chapter will do the same for gradient batch-dissonance. Recall from Definition 1.2 that gradient batch-dissonance is a quality of networks and architectures, and that architectures with lower gradient batch-dissonance will have more consistent gradient estimates for different batches. This means that the expected decrease in loss from a step for elements that aren't in the batch used to estimate the gradient used to make the step is higher. It also means that the variance between gradient estimates made with different batches will generally be lower, which means that first order optimization methods will progress more smoothly. We will establish the definition of gradient step-consistency more quantitatively and establish these results in the next section before moving on to reviewing related literature.

3.1 Quantifying Gradient Batch-Dissonance

Given two neural networks, if one network exhibits lower gradient batch-dissonance than the other within a provided context, then there are several quantifiable performance differences that will be true in expectation within that context. We will first define the concept of gradient batch-dissonance in a quantifiable way, and then proceed to prove a few more relationships that follow from this definition.

Definition 3.1 (**A Formal Definition of Gradient Batch-Dissonance**)**.** Let two neural network architectures be given, one of which (network A) exhibits lower gradient batch-dissonance than the other (network B) within some context (including a shared dataset $\mathbb{X} = \mathbb{Y}$ or two datasets $\mathbb{X} \neq \mathbb{Y}$). We will more formally define the relationship implied by this statement as follows:

For any regions $\Phi \in (\mathbb{W}_A, \|\cdot\|_2$ and $\Psi \in (\mathbb{W}_B, \|\cdot\|_2)$ that are contained within the provided context, the following relationships hold in expectation for elements ($x_i \in \mathbb{X}$, $y_p, \in \mathbb{Y}$):

$$\mathop{\mathbb{E}}_{x_i, x_j \in \mathbb{X}} \left[\frac{\|h_{\mathbb{W}_A}(\phi, x_i) - h_{\mathbb{W}_A}(\phi, x_j)\|_2}{\|h_{\mathbb{W}_A}(\phi, x_i)\|_2} \right] < \mathop{\mathbb{E}}_{y_p, y_q \in \mathbb{Y}} \left[\frac{\|h_{\mathbb{W}_B}(\psi, y_p) - h_{\mathbb{W}_B}(\psi, y_q)\|_2}{\|h_{\mathbb{W}_B}(\psi, y_p)\|_2} \right] \tag{3.1}$$

where $\phi \in \Phi$ and $\psi \in \Psi$.

Theorem 3.2 (**Expected Performance in Networks with Low Batch-Dissonance**)**.** *Let two neural network architectures be given, one of which (network A) exhibits lower gradient batch-dissonance than the other (network B) within some context (including a shared dataset $\mathbb{X} = \mathbb{Y}$ or two datasets $\mathbb{X} \neq \mathbb{Y}$). In other words, assume that equation (3.1) is true for the provided context.*

For any regions $\Phi \in (\mathbb{W}_A, \|\cdot\|_2$ and $\Psi \in (\mathbb{W}_B, \|\cdot\|_2)$ that are contained within the provided context, the following relationships hold in expectation for elements ($x_i \in \mathbb{X}$, $y_p, \in \mathbb{Y}$) and batches of the same size ($\chi_i \in B_b(\mathbb{X})$, $\gamma_p \in B_b(\mathbb{Y})$) drawn from the provided datasets:

$$\underset{\chi_i \in B_b(\mathbb{X}), x_j \in \mathbb{X}}{\mathbb{E}} \left[\frac{\left\| \overline{h_{\mathbb{W}_A}(\phi, \chi_i)} - h_{\mathbb{W}_A}(\phi, x_j) \right\|_2}{\left\| \overline{h_{\mathbb{W}_A}(\phi, \chi_i)} \right\|_2} \right]$$
$$< \underset{\gamma_p \in B_b(\mathbb{Y}), y_q \in \mathbb{X}}{\mathbb{E}} \left[\frac{\left\| \overline{h_{\mathbb{W}_B}(\psi, \gamma_p)} - h_{\mathbb{W}_B}(\psi, y_q) \right\|_2}{\left\| \overline{h_{\mathbb{W}_B}(\psi, \gamma_p)} \right\|_2} \right] \tag{3.2}$$

and

$$\underset{\chi_i, \chi_j \in B_b(\mathbb{X})}{\mathbb{E}} \left[\frac{\left\| \overline{h_{\mathbb{W}_A}(\phi, \chi_i)} - \overline{h_{\mathbb{W}_A}(\phi, \chi_j)} \right\|_2}{\left\| \overline{h_{\mathbb{W}_A}(\phi, \chi_i)} \right\|_2} \right]$$
$$< \underset{\gamma_p, \gamma_q \in B_b(\mathbb{Y})}{\mathbb{E}} \left[\frac{\left\| \overline{h_{\mathbb{W}_B}(\psi, \gamma_p)} - \overline{h_{\mathbb{W}_B}(\psi, \gamma_q)} \right\|_2}{\left\| \overline{h_{\mathbb{W}_B}(\psi, \gamma_p)} \right\|_2} \right] \tag{3.3}$$

where $\phi \in \Phi$ *and* $\psi \in \Psi$.

Furthermore, when a step is made in the direction of a batch-averaged gradient, the expected average decrease in loss for individual elements for network A is closer to the decrease in loss they would have experienced individually from a step of the same size in the direction of their own gradient than for network B. This is expressed as an inequality below:

$$\frac{\underset{\chi \in B(\mathbb{X})}{\mathbb{E}} \left[\overline{\mathcal{L}_A(\phi, \chi)} \right] - \underset{\chi \in B(\mathbb{X})}{\mathbb{E}} \left[\overline{\mathcal{L}_A(\phi - \alpha(\chi), \chi)} \right]}{\underset{x \in \mathbb{X}}{\mathbb{E}} \left[\mathcal{L}_A(\phi, x) \right] - \underset{x \in \mathbb{X}}{\mathbb{E}} \left[\mathcal{L}_A(\phi - \alpha(x), x) \right]} \tag{3.4}$$
$$> \frac{\underset{\gamma \in B(\mathbb{Y})}{\mathbb{E}} \left[\overline{\mathcal{L}_B(\psi, \gamma)} \right] - \underset{\gamma \in B(\mathbb{Y})}{\mathbb{E}} \left[\overline{\mathcal{L}_B(\psi - \beta(\gamma), \gamma)} \right]}{\underset{y \in \mathbb{Y}}{\mathbb{E}} \left[\mathcal{L}_B(\psi, y) \right] - \underset{y \in \mathbb{Y}}{\mathbb{E}} \left[\mathcal{L}_B(\psi - \beta(y), y) \right]}$$

$$\textit{where } \alpha(\chi) = \delta_1 \overline{h_{\mathbb{W}_A}(\phi, \chi)},\ \alpha(x) = \delta_1 h_{\mathbb{W}_A}(\phi, x, t), \quad (x, t) \in \mathbb{X} \times \mathbb{T}_{\mathbb{A}} \ ,$$
$$\beta(\gamma) = \delta_2 \overline{h_{\mathbb{W}_B}(\psi, \gamma)},\ \beta(x) = \delta_2 h_{\mathbb{W}_B}(\psi, y, s), \quad (y, s) \in \mathbb{Y} \times \mathbb{T}_{\mathbb{B}} \ , \tag{3.5}$$
$$0 < \delta_1 < \epsilon, \ \textit{and}\ 0 < \delta_2 < \epsilon \tag{3.6}$$

for some sufficiently small $\epsilon > 0$.

Because the expected gain from a step in the direction of a batch-averaged gradient is bounded above by the expected gain from a step of the same size in the direction of an individual gradient, this ratio is between zero and one for both networks.

Simply speaking, the relationship expressed in equation (3.4) means that the cost inflicted on network A's learning speed by dissimilarities in the input space is less for network A than it is for network B. Furthermore, network A is less likely to experience stagnation during learning that can be caused by the batch-gradient not decreasing the loss for a sizeable portion of the input space.

Proof. Assume the hypothesis. Equations 3.2 and 3.3 follow naturally from equation (3.1) because averaging either gradient estimate over more elements doesn't modify the expectation at all, and in fact significantly reduces the variance of the input to the expectation in each case. However, in order to maintain the inequality between the two networks the batch size of an element on either side must be changed in the same way.

In order to prove equation (3.4) consider the following. By the definition of the gradient we have for any point in weight space, we will use $\phi \in \Phi$ from the context for network A here, that $\|h_{\mathbb{W}_A}(\phi, x_i, t_i)\|^2 = \mathcal{L}_\phi(h_{\mathbb{W}_A}(\phi, x_i, t_i))$ the directional derivative in the direction of the gradient. Furthermore, using a first-order approximation we can estimate the change in loss from a step α (as defined in 3.4):

$$\begin{aligned}
\mathcal{L}_\mathcal{A}(\phi, x_i, t_i) - \mathcal{L}_\mathcal{A}(\phi - \alpha(x_i), x_i, t_i) &\approx \langle \alpha(x_i), h_{\mathbb{W}_A}(\phi, x_i, t_i) \rangle \\
&= \langle \delta_1 h_{\mathbb{W}_A}(\phi, x_i, t_i), h_{\mathbb{W}_A}(\phi, x_i, t_i) \rangle \\
&= \delta_1 \left\| h_{\mathbb{W}_A}(\phi, x_i, t_i) \right\|_2^2 \\
&= \delta_1 \mathcal{L}_\phi(h_{\mathbb{W}_A}(\phi, x_i, t_i)) \ . \qquad (3.7)
\end{aligned}$$

The change in overall loss from a step in the direction of a batch gradient can be estimated

in the same way:

$$\begin{aligned}\mathcal{L}_{\mathcal{A}}(\phi,\chi)-\mathcal{L}_{\mathcal{A}}(\phi-\alpha(\chi),\chi) &\approx \delta_1\mathcal{L}_{\phi}(\overline{h_{\mathbb{W}_A}(\phi,\chi)}) \\ &= \delta_1\left\|\overline{h_{\mathbb{W}_A}(\phi,\chi)}\right\|_2^2 .\end{aligned} \tag{3.8}$$

Furthermore, by Theorem 1 in [3], which proves that the expected decrease in loss for an individual element from a small step in the direction of its gradient is greater than the expected decrease in the loss evaluated using a batch from a step of the same size in the direction of the batch-averaged gradient, we have that equation (3.8) is bounded above in expectation by equation (3.7), with equality guaranteed when every input is the same. In other words:

$$\mathop{\mathbb{E}}_{x\in\mathbb{X}}\left[\mathcal{L}_A(\phi,x)-\mathcal{L}_A(\phi-\alpha(x),x)\right] \geq \mathop{\mathbb{E}}_{\chi\in B(\mathbb{X})}\left[\overline{\mathcal{L}_A(\phi,\chi)}-\overline{\mathcal{L}_A(\phi-\alpha(\chi),\chi)}\right]$$

or, equivalently,

$$\mathop{\mathbb{E}}_{x\in\mathbb{X}}\left[\|h_{\mathbb{W}_A}(\phi,x,t_x)\|_2^2\right]-\mathop{\mathbb{E}}_{\chi\in B(\mathbb{X})}\left[\left\|\overline{h_{\mathbb{W}_A}(\phi,\chi)}\right\|_2^2\right]\geq 0$$

where t_x is the label for input x. This difference is the expected efficiency loss from a given optimization step of size δ_1 caused by disagreement between gradients within the batch. To say it another way, the expected decrease in loss for a step of size δ_1 with a batch χ is less than or equal to the expected decrease in loss for a step of the same size with a batch size of one, meaning the ratio between them is less than or equal to one:

$$\frac{\mathop{\mathbb{E}}_{\chi\in B(\mathbb{X})}\left[\left\|\overline{h_{\mathbb{W}_A}(\phi,\chi)}\right\|_2^2\right]}{\mathop{\mathbb{E}}_{x\in\mathbb{X}}\left[\|h_{\mathbb{W}_A}(\phi,x,t_x)\|_2^2\right]}\leq 1 . \tag{3.9}$$

Now that the meaning of the above ratio has been established, we will proceed to demonstrate that the left side of equation (3.9) is greater for network A than for network B within the

provided context using the relationship from equation (3.2) below:

$$\frac{\mathop{\mathbb{E}}_{\chi\in B_b(\mathbb{X}),x\in\mathbb{X}}\left[\left\|\overline{h_{\mathbb{W}_A}(\phi,\chi)}-h_{\mathbb{W}_A}(\phi,x,t_x)\right\|_2\right]}{\mathop{\mathbb{E}}_{\chi\in B_b(\mathbb{X})}\left[\left\|\overline{h_{\mathbb{W}_A}(\phi,\chi)}\right\|_2\right]} < \frac{\mathop{\mathbb{E}}_{\gamma\in B_b(\mathbb{Y}),y\in\mathbb{Y}}\left[\left\|\overline{h_{\mathbb{W}_B}(\psi,\gamma)}-h_{\mathbb{W}_B}(\psi,y,t_y)\right\|_2\right]}{\mathop{\mathbb{E}}_{\gamma\in B_b(\mathbb{Y})}\left[\left\|\overline{h_{\mathbb{W}_B}(\psi,\gamma)}\right\|_2\right]} . \tag{3.2}$$

By the properties of inner products we know that

$$\left\|\overline{h_{\mathbb{W}_A}(\phi,\chi)}-h_{\mathbb{W}_A}(\phi,x,t_x)\right\|_2^2 = \left\|\overline{h_{\mathbb{W}_A}(\phi,\chi)}\right\|_2^2 - 2\left\langle\overline{h_{\mathbb{W}_A}(\phi,\chi)},h_{\mathbb{W}_A}(\phi,x,t_x)\right\rangle + \|h_{\mathbb{W}_A}(\phi,x,t_x)\|_2^2 .$$

Using this to expand the terms on each side of equation (3.2) gives us the following relationship:

$$\mathop{\mathbb{E}}_{\chi\in B_b(\mathbb{X}),x\in\mathbb{X}}\left[\frac{\left\|\overline{h_{\mathbb{W}_A}(\phi,\chi)}\right\|_2^2 - 2\left\langle\overline{h_{\mathbb{W}_A}(\phi,\chi)},h_{\mathbb{W}_A}(\phi,x,t_x)\right\rangle + \|h_{\mathbb{W}_A}(\phi,x,t_x)\|_2^2}{\left\|\overline{h_{\mathbb{W}_A}(\phi,\chi)}\right\|_2^2}\right]$$
$$< \mathop{\mathbb{E}}_{\gamma\in B_b(\mathbb{Y}),y\in\mathbb{Y}}\left[\frac{\left|\left\|\overline{h_{\mathbb{W}_B}(\psi,\gamma)}\right\|_2^2 - 2\left\langle\overline{h_{\mathbb{W}_B}(\psi,\gamma)},h_{\mathbb{W}_B}(\psi,y,t_y)\right\rangle + \|h_{\mathbb{W}_B}(\psi,y,t_y)\|_2^2\right|}{\left\|\overline{h_{\mathbb{W}_B}(\psi,\gamma)}\right\|_2^2}\right] .$$

Consider the following re-arrangement of terms:

$$\mathop{\mathbb{E}}_{\chi\in B_b(\mathbb{X}),x\in\mathbb{X}}\left[\frac{\left(\left\|\overline{h_{\mathbb{W}_A}(\phi,\chi)}\right\|_2^2+\|h_{\mathbb{W}_A}(\phi,x,t_x)\|_2^2\right)-2\left\langle\overline{h_{\mathbb{W}_A}(\phi,\chi)},h_{\mathbb{W}_A}(\phi,x,t_x)\right\rangle}{\left\|\overline{h_{\mathbb{W}_A}(\phi,\chi)}\right\|_2^2}\right]$$
$$<\mathop{\mathbb{E}}_{\gamma\in B_b(\mathbb{Y}),y\in\mathbb{Y}}\left[\frac{\left(\left\|\overline{h_{\mathbb{W}_B}(\psi,\gamma)}\right\|_2^2+\|h_{\mathbb{W}_B}(\psi,y,t_y)\|_2^2\right)-2\left\langle\overline{h_{\mathbb{W}_B}(\psi,\gamma)},h_{\mathbb{W}_B}(\psi,y,t_y)\right\rangle}{\left\|\overline{h_{\mathbb{W}_B}(\psi,\gamma)}\right\|_2^2}\right]. \tag{3.10}$$

Note that because the sum of the two norms in the numerator is strictly positive, minimization can only occur by convergence of the inner product to the average of the two norms. In other words, the inner product for network A must be closer in expectation to the average of the squared gradient magnitudes than for network B. We can use this as follows. Equation (3.1) implies that

$$\mathop{\mathbb{E}}_{\chi\in B_b(\mathbb{X}),x\in\mathbb{X}}\left[\frac{\left(\left\|\overline{h_{\mathbb{W}_A}(\phi,\chi)}\right\|_2^2+\|h_{\mathbb{W}_A}(\phi,x,t_x)\|_2^2\right)}{2\left\|\overline{h_{\mathbb{W}_A}(\phi,\chi)}\right\|_2^2}-\frac{\left\langle\overline{h_{\mathbb{W}_A}(\phi,\chi)},h_{\mathbb{W}_A}(\phi,x,t_x)\right\rangle}{\left\|\overline{h_{\mathbb{W}_A}(\phi,\chi)}\right\|_2^2}\right]$$
$$<\mathop{\mathbb{E}}_{\gamma\in B_b(\mathbb{Y}),y\in\mathbb{Y}}\left[\frac{\left(\left\|\overline{h_{\mathbb{W}_B}(\psi,\gamma)}\right\|_2^2+\|h_{\mathbb{W}_B}(\psi,y,t_y)\|_2^2\right)}{2\left\|\overline{h_{\mathbb{W}_B}(\psi,\gamma)}\right\|_2^2}-\frac{\left\langle\overline{h_{\mathbb{W}_B}(\psi,\gamma)},h_{\mathbb{W}_B}(\psi,y,t_y)\right\rangle}{\left\|\overline{h_{\mathbb{W}_B}(\psi,\gamma)}\right\|_2^2}\right],$$

but, because

$$\mathop{\mathbb{E}}_{\chi\in B_b(\mathbb{X})}\left[\left\|\overline{h_{\mathbb{W}_A}(\phi,\chi)}\right\|_2^2\right]=\mathop{\mathbb{E}}_{\chi\in B_b(\mathbb{X})}\left[\|h_{\mathbb{W}_A}(\phi,x,t_x)\|_2^2\right]$$

we can simplify this as

$$\mathop{\mathbb{E}}_{\chi\in B_b(\mathbb{X}),x\in\mathbb{X}}\left[\frac{2\left\|h_{\mathbb{W}_A}(\phi,x,t_x)\right\|_2^2}{2\left\|\overline{h_{\mathbb{W}_A}(\phi,\chi)}\right\|_2}-\frac{\left\langle\overline{h_{\mathbb{W}_A}(\phi,\chi)},h_{\mathbb{W}_A}(\phi,x,t_x)\right\rangle}{\left\|\overline{h_{\mathbb{W}_A}(\phi,\chi)}\right\|_2^2}\right]$$
$$<\mathop{\mathbb{E}}_{\chi\in B_b(\mathbb{Y}),y\in\mathbb{Y}}\left[\frac{2\left\|h_{\mathbb{W}_B}(\psi,y,t_y)\right\|_2^2}{2\left\|\overline{h_{\mathbb{W}_B}(\psi,\gamma)}\right\|_2}-\frac{\left\langle\overline{h_{\mathbb{W}_B}(\psi,\gamma)},h_{\mathbb{W}_B}(\psi,y,t_y)\right\rangle}{\left\|\overline{h_{\mathbb{W}_B}(\psi,\gamma)}\right\|_2^2}\right].$$

Define τ_1 as the numerator of the above expectation for network A and τ_2 as the numerator of the above expectation for network B. Then consider following substitution:

$$\mathop{\mathbb{E}}_{\chi\in B_b(\mathbb{X}),x\in\mathbb{X}}\left[\frac{\left\|\overline{h_{\mathbb{W}_A}(\phi,\chi)}\right\|_2^2-2\left\langle\overline{h_{\mathbb{W}_A}(\phi,\chi)},h_{\mathbb{W}_A}(\phi,x,t_x)\right\rangle+\left\|h_{\mathbb{W}_A}(\phi,x,t_x)\right\|_2^2}{\left\|\overline{h_{\mathbb{W}_A}(\phi,\chi)}\right\|_2^2}\right]$$
$$=\mathop{\mathbb{E}}_{\chi\in B_b(\mathbb{X}),x\in\mathbb{X}}\left[\frac{\left\|\overline{h_{\mathbb{W}_A}(\phi,\chi)}\right\|_2^2-2\left\langle\overline{h_{\mathbb{W}_A}(\phi,\chi)},h_{\mathbb{W}_A}(\phi,x,t_x)\right\rangle-2\tau_1+2\tau_1+\left\|h_{\mathbb{W}_A}(\phi,x,t_x)\right\|_2^2}{\left\|\overline{h_{\mathbb{W}_A}(\phi,\chi)}\right\|_2^2}\right]$$
$$=\mathop{\mathbb{E}}_{\chi\in B_b(\mathbb{X}),x\in\mathbb{X}}\left[\frac{\left\|\overline{h_{\mathbb{W}_A}(\phi,\chi)}\right\|_2^2-2\left\|h_{\mathbb{W}_A}(\phi,x,t_x)\right\|_2^2+2\tau_1+\left\|h_{\mathbb{W}_A}(\phi,x,t_x)\right\|_2^2}{\left\|\overline{h_{\mathbb{W}_A}(\phi,\chi)}\right\|_2^2}\right]$$
$$=\mathop{\mathbb{E}}_{\chi\in B_b(\mathbb{X}),x\in\mathbb{X}}\left[\frac{\left\|\overline{h_{\mathbb{W}_A}(\phi,\chi)}\right\|_2^2-\left\|h_{\mathbb{W}_A}(\phi,x,t_x)\right\|_2^2+2\tau_1}{\left\|\overline{h_{\mathbb{W}_A}(\phi,\chi)}\right\|_2^2}\right].$$

We already know that

$$\mathop{\mathbb{E}}_{\chi\in B_b(\mathbb{X}),x\in\mathbb{X}}\left[\frac{\tau_1}{\left\|\overline{h_{\mathbb{W}_A}(\phi,\chi)}\right\|_2^2}\right]<\mathop{\mathbb{E}}_{\gamma\in B_b(\mathbb{Y}),y\in\mathbb{Y}}\left[\frac{\tau_2}{\left\|\overline{h_{\mathbb{W}_B}(\psi,\gamma)}\right\|_2^2}\right],$$

therefore, we can make a corresponding replacement with τ_2 for network B, and obtain the

following relationship:

$$\underset{\chi\in B_b(\mathbb{X}),x\in\mathbb{X}}{\mathbb{E}}\left[\frac{\left\|\overline{h_{\mathbb{W}_A}(\phi,\chi)}\right\|_2^2-\|h_{\mathbb{W}_A}(\phi,x,t_x)\|_2^2-2\tau_1}{\left\|\overline{h_{\mathbb{W}_A}(\phi,\chi)}\right\|_2^2}\right]$$
$$<\underset{\gamma\in B_b(\mathbb{Y}),y\in\mathbb{Y}}{\mathbb{E}}\left[\frac{\left\|\overline{h_{\mathbb{W}_B}(\psi,\gamma)}\right\|_2^2-\|h_{\mathbb{W}_B}(\psi,y,t_y)\|_2^2-2\tau_2}{\left\|\overline{h_{\mathbb{W}_B}(\psi,\gamma)}\right\|_2^2}\right].$$

This combined with equations 3.7 and 3.8 gives us the desired result:

$$\underset{\chi\in B_b(\mathbb{X}),x\in\mathbb{X}}{\mathbb{E}}\left[\frac{(\mathcal{L}_{\mathcal{A}}(\phi,\chi)-\mathcal{L}_{\mathcal{A}}(\phi-\alpha,\chi))-(\mathcal{L}_{\mathcal{A}}(\phi,x,t_x)-\mathcal{L}_{\mathcal{A}}(\phi-\alpha,x,t_x))-2\tau_1}{\left\|\overline{h_{\mathbb{W}_A}(\phi,\chi)}\right\|_2^2}\right]$$
$$<\underset{\gamma\in B_b(\mathbb{Y}),y\in\mathbb{Y}}{\mathbb{E}}\left[\frac{(\mathcal{L}_{\mathcal{B}}(\psi,\gamma)-\mathcal{L}_{\mathcal{B}}(\psi-\beta,\gamma))-(\mathcal{L}_{\mathcal{B}}(\psi,y,t_y)-\mathcal{L}_{\mathcal{B}}(\psi-\beta,x,t_x))-2\tau_2}{\left\|\overline{h_{\mathbb{W}_B}(\psi,\gamma)}\right\|_2^2}\right]$$

which means that

$$\frac{\underset{\chi\in B_b(\mathbb{X})}{\mathbb{E}}[\mathcal{L}_{\mathcal{A}}(\phi,\chi)-\mathcal{L}_{\mathcal{A}}(\phi-\alpha,\chi)]}{\underset{(x,t_x)\in\mathbb{X}\times\mathbb{T}}{\mathbb{E}}[\mathcal{L}_{\mathcal{A}}(\phi,x,t_x)-\mathcal{L}_{\mathcal{A}}(\phi-\alpha,x,t_x)]}$$
$$>\frac{\underset{\gamma\in B_b(\mathbb{Y})}{\mathbb{E}}[\mathcal{L}_{\mathcal{B}}(\psi,\gamma)-\mathcal{L}_{\mathcal{B}}(\psi-\beta,\gamma)]}{\underset{(y,t_y)\in\mathbb{Y}\times\mathbb{T}}{\mathbb{E}}[\mathcal{L}_{\mathcal{B}}(\phi,y,t_y)-\mathcal{L}_{\mathcal{B}}(\psi-\beta,y,t_y)]}.$$

In other words, the cost inflicted on network A's learning speed by dissimilarities in the input space is less for network A than it is for network B. Note that the batch size b is in this expression. The cost inflicted on individual gain from each step increases with batch size at the cost of decreased variance between batch gradient estimates. □

The result from this theorem about the decrease in benefit from each step on a per-element basis is important. It demonstrates that there is an important trade-off between

the expected per-element decrease in loss per step and the amount of variance experienced between batch-averaged gradient estimates.

While the variance between batch-averaged gradient estimates scales predictably in proportion with $\frac{1}{\sqrt{b}}$ for every network, some networks perform much better at the other end of the trade-off. Specifically, as for network B that was used in the proof of theorem 3.2, some networks will experience a greater proportional decrease in per-element loss decrease from batches of the same size. Because of this, for some networks it may be impossible to find a batch size that is both large enough to avoid instability between steps in SGD and is still small enough that learning does not stagnate due to a lack of variance and progress for individual classes and inputs.

3.2 Gradient Conditioning with Respect to $\mathbb{X}$

Theorem 3.2 gave us some powerful insights into the learning performance of networks with low gradient batch-dissonance. However, each of the results from Theorem 3.2 in the previous section relied heavily upon the expectation found in equation (3.1). Naturally, each of the resulting relationships established were also proven only in expectation. This allowed us to prove very general statements about the performance differences that can be expected when a difference in gradient batch-dissonance is observed within a given context. However, this approach also allowed us to ignore what could amount to a very large amount of variance in actual performance.

Consider Equation (3.2), shown here:

$$\begin{aligned} & \mathop{\mathbb{E}}_{\chi_i \in B_b(\mathbb{X}), x_j \in \mathbb{X}} \left[\frac{\left\| \overline{h_{\mathbb{W}_A}(\phi, \chi_i)} - h_{\mathbb{W}_A}(\phi, x_j) \right\|_2}{\left\| \overline{h_{\mathbb{W}_A}(\phi, \chi_i)} \right\|_2} \right] \\ < & \mathop{\mathbb{E}}_{\gamma_p \in B_b(\mathbb{Y}), y_q \in \mathbb{X}} \left[\frac{\left\| \overline{h_{\mathbb{W}_B}(\psi, \gamma_p)} - h_{\mathbb{W}_B}(\psi, y_q) \right\|_2}{\left\| \overline{h_{\mathbb{W}_B}(\psi, \gamma_p)} \right\|_2} \right] . \end{aligned} \tag{3.2}$$

Depending on the dataset or datasets chosen, the variance on the term within each ex-

pectation could be extremely large. Furthermore, this statement relies upon a context being defined for equation (3.1) that could include only very limited regions in the weight space of each architecture. Theorem 3.2 might be useful for establishing the expected performance of more than two networks if a reliable context were established, for example evaluating the performance of several networks on the same dataset in a small region around a local minimizer. However, it falls far short of being applicable to the wide set of circumstances that would be required for it to be used in the loss function of an architecture search, or any of the other purposes established in chapter 1.

The issue preventing the quantity in each expectation in Equation (3.1)

$$\frac{\|h_{\mathbb{W}_A}(\phi, x_i) - h_{\mathbb{W}_A}(\phi, x_j)\|_2}{\|h_{\mathbb{W}_A}(\phi, x_i)\|_2} \tag{3.11}$$

from being used under more generic circumstances is variance introduced by the samples $x_i, x_j \in \mathbb{X}$. In other words, it is not normalized for dissimilarity between the two inputs chosen. Unfortunately, we cannot simply normalize using the distance between the two inputs. Because the inputs could be so far apart in weight space, there is no reason to believe that the distance between them actually has any reliable correlation with the differences between the gradient estimates.

However, if the distance between the chosen samples was small enough we could normalize using the distance between them (relative to their magnitudes). Normalizing the gradient step-consistency quantity in section 2.2 was very straightforward, because the step in weight space between adjacent gradient estimates is typically very small in application already, and it is easy to reduce the learning rate if needed. However, the discrete nature of the subspace in $\mathbb{X}$ that can be paired with valid labels makes it impossible to reliably sample paired elements that are close enough for this use case. Instead of using two known samples from the input space it will be necessary to measure the change in the gradient across a small perturbation to a single input, with the assumption that the same label is valid for the perturbed input. In this case the magnitude of the perturbation (relative to the magnitude of the input being

perturbed) can be easily controlled to maximize applicability of the resulting metric across different datasets. This modification results in the quantity below:

$$\frac{\|h_{\mathbb{W}_A}(\phi, x) - h_{\mathbb{W}_A}(\phi, x+\delta)\|_{\mathbb{W}}}{\|h_{\mathbb{W}_A}(\phi, x)\|_{\mathbb{W}}} \cdot \frac{\|x\|_{\mathbb{X}}}{\|\delta\|_{\mathbb{X}}} \ . \tag{3.12}$$

As it did in section 2.2 for the related equations on gradient step-consistency, the required normalization of the quantity measured in Theorem 3.2 resulted in the quantity measured by the relative condition number as it would be used for measuring changes in the gradient caused by differences in the input. We define the relative condition number for this circumstance below.

Definition 3.3 (Relative Condition Number with respect to $\mathbb{X}$)**.** Let an architecture $g : ((\mathbb{R}^m, ||\cdot||_{\mathbb{W}}), (\mathbb{R}^k, ||\cdot||_{\mathbb{X}})) \to \mathbb{R}^j$ and a loss function $f : (\mathbb{R}^j, T^h) \to \mathbb{R}$ be given. Using the notation of Definition 1.3, the relative gradient condition number $\kappa_{h_{\mathbb{W}},\mathbb{X}}(w, x, t), w \in \mathbb{W}, x \in \mathbb{X}, t \in \mathbb{T}^h$, with respect to the input space $\mathbb{X}$ is defined as follows:

$$\kappa_{h_{\mathbb{W}},\mathbb{X}}(w, x, t) = \hat{\kappa}_{h_{\mathbb{W}},\mathbb{X}}(w, x, t) \cdot \frac{||x||_{\mathbb{X}}}{||h_{\mathbb{W}}(w, x, t)||_{\mathbb{W}}} \tag{3.13}$$

where

$$\hat{\kappa}_{h_{\mathbb{W}},\mathbb{X}}(w, x, t) = \lim_{\epsilon \to 0} \sup_{||\delta||_{\mathbb{X}} < \epsilon} \frac{||h_{\mathbb{W}}(w, x+\delta, t) - h_{\mathbb{W}}(w, x, t)||_{\mathbb{W}}}{||\delta||_{\mathbb{X}}} = ||J_{h_{\mathbb{W}},\mathbb{X}}(w, x, t)||_{\mathbb{X},\mathbb{W}} \ ,$$

$J_{h_{\mathbb{W}},\mathbb{X}}(w)$ is the Jacobian of $h_{\mathbb{W}}$ with respect to $\mathbb{X}$ at w and $||\cdot||_{\mathbb{X},\mathbb{W}}$ is the induced norm on transformations between $\mathbb{X}$ and $\mathbb{W}$. The second equality in each case holds if and only if $J_{h_{\mathbb{W}},\mathbb{X}}(w, x, t)$ is defined at (w, x, t).

By theorem 1.6, if $h_{\mathbb{W}}(w, x, t)$ is differentiable with respect to $\mathbb{X}$ at $(w, x, t) \in \mathbb{W} \times \mathbb{X} \times \mathbb{T}^h$, then an $\epsilon > 0$ exists such that

$$\kappa_{h_{\mathbb{W}},\mathbb{X}}(w, x, t) + o(\epsilon) \geq \frac{||h_{\mathbb{W}}(w, x+\delta, t) - h_{\mathbb{W}}(w, x, t)||_{\mathbb{W}}||x||_{\mathbb{X}}}{||\delta||_{\mathbb{X}}||h_{\mathbb{W}}(w, x, t)||_{\mathbb{W}}} \tag{3.14}$$

for any $\delta \in \mathbb{X}$ where $||\delta||_{\mathbb{W}} < \epsilon$.

This allows us to define a viable metric for gradient batch-dissonance that does not rely on an expectation, a limit, or a supremum: the approximate best lower bound on $\kappa_{h_{\mathbb{W}},\mathbb{X}}$.

Definition 3.4 (Approximate Best Lower Bound on $\kappa_{h_{\mathbb{W}},\mathbb{X}}$)**.** Let $\epsilon > 0$ and $n \in \mathbb{N}$ be given and let $B_\epsilon(0) = \{p \in \mathbb{X} : 0 < ||p||_{\mathbb{X}} < \epsilon\}$. Then we define our approximate best lower bound on $\kappa_{h_{\mathbb{W}},\mathbb{X}}(w, x, t)$ as

$$\overset{\geq}{\approx}_{h_{\mathbb{W}},\mathbb{X}}(w, x, t) := \max_{\delta_k \in B_\epsilon(0), 0 \leq k < n} \frac{||h_{\mathbb{W}}(w, x + \delta_k, t) - h_{\mathbb{W}}(w, x, t)||_{\mathbb{W}} ||x||_{\mathbb{X}}}{||\delta_k||_{\mathbb{X}} ||h_{\mathbb{W}}(w, x, t)||_{\mathbb{W}}} \tag{3.15}$$

with an accompanying average defined using a batch of size b as follows:

$$\overline{\overset{\geq}{\approx}_{h_{\mathbb{W}},\mathbb{X}}(w, \chi)} := \frac{1}{m} \sum_{(x,t)_i \in \chi, i < m} \left(\max_{\delta \in \gamma_n} \frac{||h_{\mathbb{W}}(w, x + \delta, t) - h_{\mathbb{W}}(w, x, t)||_{\mathbb{W}} ||x||_{\mathbb{X}}}{||\delta||_{\mathbb{X}} ||h_{\mathbb{W}}(w, x, t)||_{\mathbb{W}}} \right) \tag{3.16}$$

$$\text{where} \quad \gamma_n = \{\gamma_k \in B_\epsilon(0) | 0 \leq k < n\}$$

and $m, n \leq b$ for optimal use of parallelization.

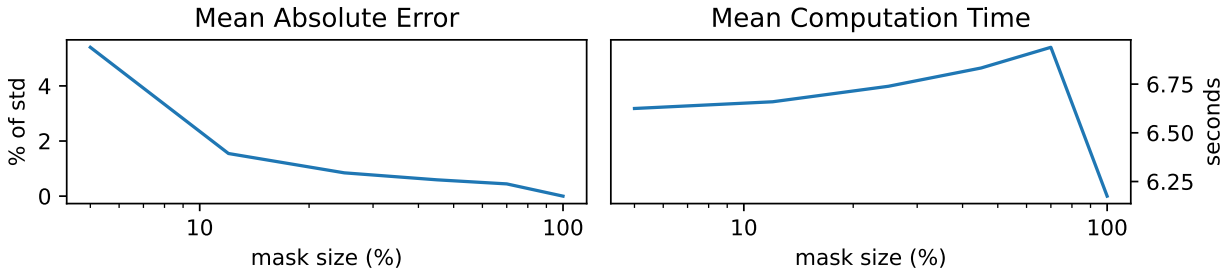


Figure 3.1: Mask Effectiveness for Gradient Conditioning w.r.t. $\mathbb{X}$

If a batch size $b = \max(m, n)$ is used and the same set Γ_n is used for every $(x, t)_i \in \chi$, then this metric can be calculated in n iterations, with each iteration requiring a forward and backward pass through the network. However, because two sets of pre-averaged batch gradients must be used simultaneously it can nearly double the worst-case memory usage that occurs during a typical optimization step with the same batch-size if implemented without the memory-reducing random masking discussed in section 1.6. However, as can be seen in figure 3.1, the temporal savings achieved in the norm calculation from random masking

was not sufficient to overcome the additional cost incurred by our non-parallelized masking implementation.

3.3 Generating Targeted Input Perturbation Samples

Previously, in section 2.3, we discussed how random directions in weight space are frequently degenerate, meaning changes along those directions have very little impact on the performance of the network. This may also true in the input space for networks that have been partially or fully trained. For example, consider a simple image classification network. If the network is able to generalize and correctly identify a given class of images, it must also be able to ignore changes to each image along directions that don't change the classification. This expected insensitivity means that the lower bound on the maximum in equation (3.16) could converge very slowly, possibly requiring a much larger sample size than is desireable.

In section 2.3, we described how we can avoid sampling degenerate directions in the weight space by sampling perturbation directions from the individual gradient estimates, $h_{\mathbb{W}}(w, x_i, t_i)$, in the evaluating batch. Directions sampled in this way are less likely to be degenerate and have a higher expected change in loss than samples drawn via a fully random process. A larger expected change in loss means that they will also have a higher expected change in the gradient (which depends directly on the loss via the chain rule), thus increasing the initial rate of convergence for the maximum in equation (2.5).

We can decrease the likelihood of sampling degenerate directions in the input space in a similar way by sampling from from a batch of gradient estimates with respect to the input space. A gradient taken with respect to a given input maximizes the change in loss for a sufficiently small perturbation of fixed size for that input. Furthermore, this effect generalizes to a lesser degree to other inputs from the same class, resulting in an expected maximum change in loss that is greater than for fully random samples in partially or fully converged networks. This generalization effect has been demonstrated in studies on adversarial attacks [10, 21, 27], where it has been found that adversarially targeted noise generalizes across

inputs in the same class, and even between networks with different architectures.

We will generate samples using gradients with respect to the input as detailed in the following definition:

Definition 3.5 (**Targeted Sampling Technique for Input Perturbation Directions**)**.** Let a batch $\chi \in B_b(\mathbb{X})$ and a sample size $n < b$ be given. When generating input perturbation direction samples, δ_i, where it is desireable to maximize the change in loss or in the gradient (such as δ in equations 3.15 and 3.16), the samples will be drawn from the set

$$\left\{ \frac{\eta \ \zeta \ h_{\mathbb{X}}(w, x_i, t_i)}{\|h_{\mathbb{X}}(w, x_i, t_i)\|_{\mathbb{X}}} \mid \zeta \sim \mathcal{U}(\{-1, 1\}),\ h_{\mathbb{X}}(w, x_i, t_i) \in h_{\mathbb{X}}(w, \chi) \right\} \tag{3.17}$$

where $\eta \in \mathbb{R}$ is the desired scale. Unless otherwise specified we will use $\eta = .001$.

We can efficiently calculate up to b targeted samples simply by adding the input layer to the backward pass used to calculate the samples $h_{\mathbb{W}}(w, x, t) \in h_{\mathbb{W}}(w, \chi)$ that are already used in equation (3.16) —the input layer is computationally identical to any other layer in the backward pass, and increases the expense at the same rate that adding additional learnable parameters would. We performed a test comparing this targeted sampling method to samples with elements drawn from a uniform distribution with the same mean and standard deviation as the elements of the source batch and scaled down by the same scaling factor $\eta = .001$. The results of this test can be seen in figure 3.2. Unlike in section 2.3, the targeted sampling technique had no apparent benefit in maximizing the lower bound on the condition number in equation (3.16). We also added an interpolation toward another random element of the batch as a third direction sample option and it also performed identically to random sampling in the 11-layer convolutional network. Because generating the targeted samples is so cheap however we will continue using that method in case it has a larger impact near the end of training or in deeper networks.

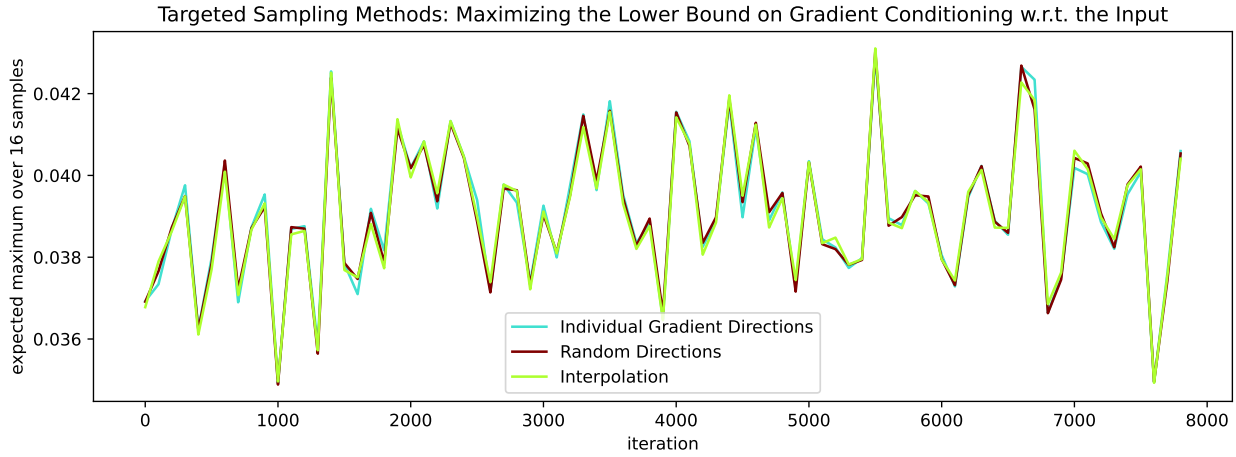


Figure 3.2: Targeted vs Random Sampling in $\mathbb{X}$. During the training of an 11 layer convolutional network (with normalization but without skip connections), 64 input perturbation directions were generated via the sampling methods described in section 3.3. Bootstrapping was used to estimate an expected maximum over a sample size of 16.

Using the input-gradient targeted sampling technique, we measured the gradient conditioning with respect to the input space for several networks throughout a full training run. The results are displayed in figure 3.3.

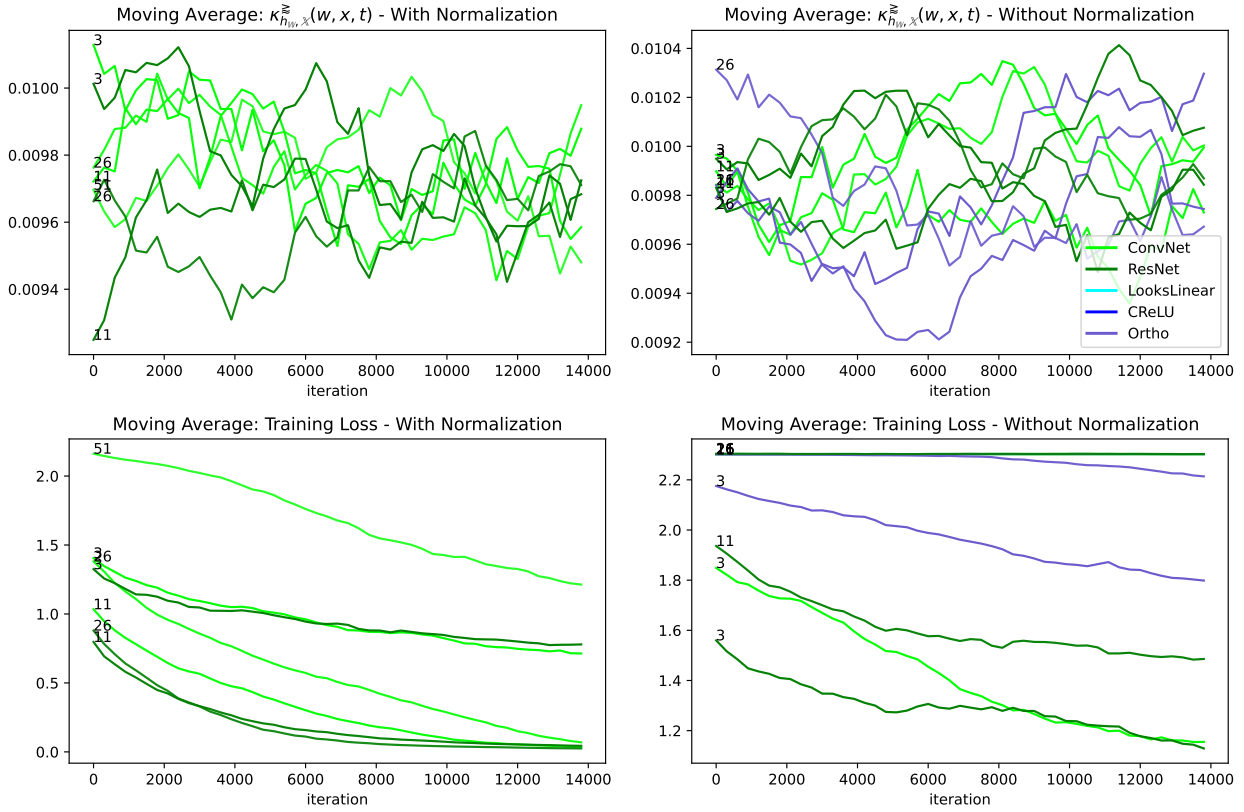


Figure 3.3: The Gradient Conditioning w.r.t. the Input Space (see Definition 3.16) was measured throughout training for several different types of architectures. No apparent correlation is apparent between learning effectiveness and the Gradient Conditioning w.r.t. the Input Space. In every case the metric was extremely consistent (notice the scale in the upper two graphs).
The architectures tested included fully connected networks with orthogonal initialization and the ReLU or CReLU activation function, the Looks-Linear architecture (see Definition 3.9), and convolutional networks with and without residual connections. The label on each line is the number of layers in the architecture.

From figure 3.3 above it seems that there is very little correlation between training success and measurements of the lower bound on our relative gradient condition number with respect to the input space. We had in fact expected plots that looked similar to this from the targeted sampling test performed previously. However, in the figure below it seems that there may be a correlation between our lower gradient conditioning bound and the slope of the loss function. It is difficult to tell however because the scale on each gradient conditioning axis is quite small.

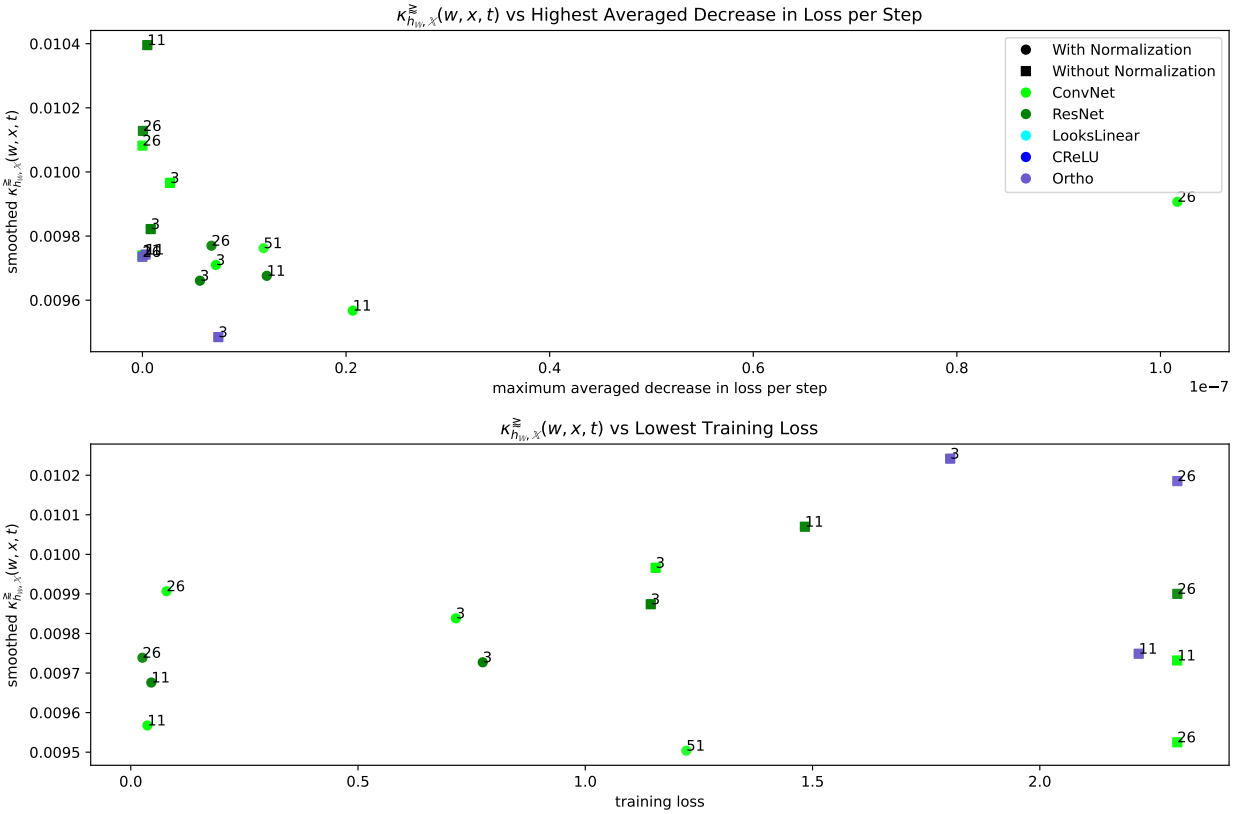


Figure 3.4: The Gradient Conditioning w.r.t. the Input Space was measured throughout training for several different types of architectures at the point of steepest descent as well as at the point with the lowest loss (smoothed over a few hundred iterations in each case) and plotted as a point for each network. Excluding the outlier ConvNet26, there is a strong correlation between the fastest rate of descent and the gradient conditioning w.r.t. the input at that point.
The architectures tested include fully connected networks with orthogonal initialization and the ReLU or CReLU activation function, the Looks-Linear architecture (see Definition 3.9), and convolutional networks with and without residual connections. The label on each point is the number of layers in the architecture.

While the correlation between gradient conditioning in the weight space and the highest average decrease in loss (a moving average over 20 samples) is not as pronounced as we would like, it does seem clear that exceptionally poor conditioning results in a near complete lack of convergence.

3.4 Literature and Metrics on Gradient Batch-Dissonance

Section 2.4 on literature relating to conditioning in the weight space was relatively short. While there are dozens of papers related to smoothness in the loss surface and gradient

step-conditioning, the methods used in those papers are generally very expensive to use and are not easy to compress into a metric that might be used to direct an architecture search. Fortunately, however, there have been several recent papers related to gradient batch-dissonance and conditioning in the input that are easily applicable to our own empirical study. Specifically, several papers introduce strongly related metrics on individual or batch-gradient variance that are easily adapted for our own tests. Because of that, the literature review for this chapter has been broken down into several sections, and each one includes some of our own results in addition to an actual review of the paper under consideration.

Papers relating to gradient batch-dissonance and conditioning with respect to the input space generally fall into two broad categories. The first category is primarily concerned with the ability of neural networks to generalize in the input space and with their robustness against input modification adversarial attacks [10, 21, 27]. The second category is more directly related to this thesis, in that they are concerned with the amount of variance that occurs between gradient estimates and the effect this has on training. Several different metrics have been proposed by these papers to measure gradient variance, and each paper establishes theorems or numerical correlations between their metric and learning ability.

These papers have introduced several different terms to describe the same issue. These include: gradient confusion, gradient step-consistency, gradient diversity, shattered gradients, gradient whitening, gradient coherence, etc. Some of these refer to batch-dissonance or variance between batch gradient estimates and some refer to variance between individual estimates within each batch. To reduce confusion we will consistently use the phrase *gradient batch-dissonance*. Unless otherwise specified, this refers to batch-dissonance between individual gradient estimates at a fixed point in weight space. The phrase *batch gradient batch-dissonance* will be used when referring to batches.

A metric called β-smoothness was recently introduced by Yang et al. in [10] that is almost identical to the absolute condition number in Definition 3.4, except it takes the maximum over pairs of distinct inputs instead of perturbing individual ones. They note that similar

metrics have been seen in the literature several times previously, and that their definition can be interpreted as the Lipschitz bound on $h_{\mathbb{W}}$. Furthermore, they demonstrate that it is an important indicator of robustness against adversarial attacks.

Another paper, the one that introduced the phrases "gradient whitening" and "shattered gradients", provided the original motivation for this thesis. In their paper "The Shattered Gradients Problem: If resnets are the answer, then what is the question?" [1] Balduzzi et al. strike particularly closely to our topic. Balduzzi et al. clearly demonstrate something they call gradient whitening, a phenomenon where changes in the derivative of the loss function caused by small changes to the input lose any visible structure with respect to the size of the change in the input. Because this loss of structure makes gradient samples look like white noise and they name it gradient whitening.

After defining a metric to measure gradient whitening, Balduzzi et al. found two notable architectural traits that significantly reduce gradient whitening and allow for the training of much deeper networks. The first is skip connections, and the second is use of the activation function CReLU with a specific initialization pattern that minimizes the discontinuity in the derivative of CReLU. We consider both more in depth in our first subsection on literature related to conditioning in $\mathbb{X}$ below.

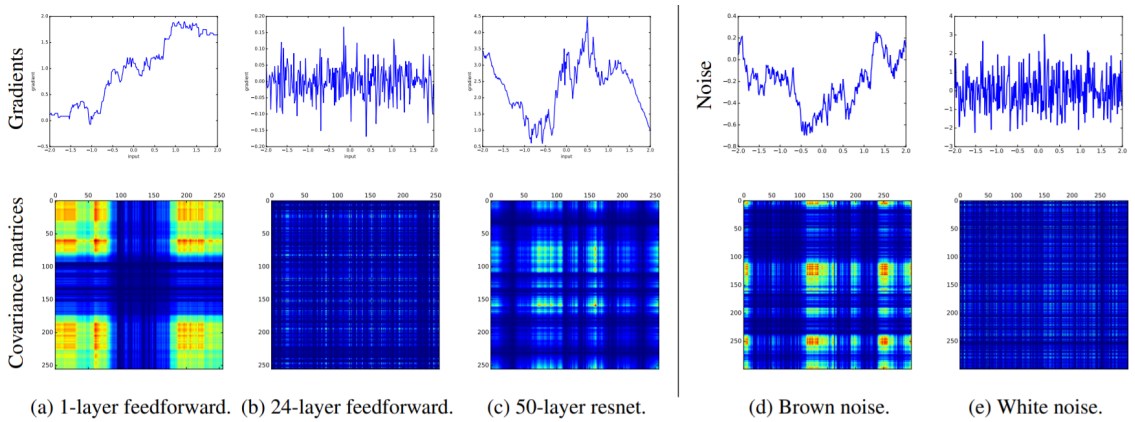
3.5 Shattered Gradients, the Gradient Rank Metric, and CReLU

In their paper "The Shattered Gradients Problem: If resnets are the answer, then what is the question?" [1], Balduzzi et al. question why, when gradient magnitudes are well managed by careful initialization and batch normalization, that architectures containing skip connections perform so much better in practice than feedforward networks.

To answer this question they investigate the condition of the loss function with respect to the input space and subsequently define a metric to measure variance in the partial derivatives of the loss function with respect to the input space ($h_{\mathbb{X}}(w, x, t)$ as defined in Definition 1.3). We are primarily concerned with the conditioning of the gradient $h_{\mathbb{W}}(w, x, t)$

with respect to the input space, but these are directly related. Namely, if the loss function itself is poorly conditioned with respect to the input, then the gradient $h_{\mathbb{W}}(w, x, t)$ will likely also be poorly conditioned with respect to the input.

Among other things, in order to empirically demonstrate the conditioning of ResNet and standard feedforward architectures of various depths, they construct several architectures, all of which map $\mathbb{R}^1 \to \mathbb{R}^1$, such that $h_{\mathbb{X}}(w, x, t) \in \mathbb{R}^1$, and plot $h_{\mathbb{X}}(w, x, t)$ vs x for $-2 \leq x \leq 2$ giving the results shown in figure 3.5.



(a) 1-layer feedforward. (b) 24-layer feedforward. (c) 50-layer resnet. (d) Brown noise. (e) White noise.

Figure 3.5: Using a small toy network mapping a 1-dimensional input to a 1-dimensional output, Balduzzi et al. demonstrated that gradients lose structure in deep feedforward networks without skip connections. This gradient decay is much less pronounced in networks containing residual connections. Figure found in [1] under the name Figure 1.

For the tested networks, they note that $h_{\mathbb{X}}(w, x, t)$ resembles brown noise in the shallow networks and in ResNet architectures, and resembles white noise in the non-shallow feedforward networks without skip connections. In other words, they show that $h_{\mathbb{X}}(w, x, t)$ is poorly conditioned in their non-shallow architectures without skip connections. They name this phenomenon *gradient whitening*, and refer to it as the *shattered gradients problem*. Furthermore, they note that gradient whitening is further reduced by use of batch normalization.

Thereafter they propose using a metric known as the *relative effective rank* to measure dissimilarity between values of $h_{\mathbb{X}}(w, x, t)$ for each element of a mini-batch χ. We will use the name *relative gradient rank* to refer to the relative effective rank as used in this context and define this below.

Definition 3.6. The effective, numerical, or stable rank of a matrix A is defined as follows:

$$R(A) = \frac{\|A\|_F^2}{\|A\|_2^2} = \frac{tr(A^T A)}{\|A\|_2^2} \ . \tag{3.18}$$

$R(A)$ is bounded above by the rank of A. The *effective rank* of A *relative* to Y is

$$\frac{R(A)}{R(Y)} \tag{3.19}$$

and it compares the second moments of A and Y.

Definition 3.7 (Relative Gradient Rank)**.** Let an architecture $g : ((\mathbb{R}^m, ||\cdot||_{\mathbb{W}}), (\mathbb{R}^k, ||\cdot||_{\mathbb{X}})) \to \mathbb{R}^j$ and a loss function $f : (\mathbb{R}^j, T^h) \to \mathbb{R}$ be given. Using the notation given in Definition 1.3, and the effective rank R from 3.18, we define the *relative gradient rank* $\Delta_{\mathbb{X}} R(h, w, \chi)$ as follows:

$$\begin{aligned} &\Delta_{\mathbb{X}} R(h, w, \chi) := \frac{R(A(h, w, \chi))}{R(Y)} \\ &w \in \mathbb{W}, \quad \chi \in B_b, \quad \text{and} \quad Y \in \{\Lambda \in \mathbb{R}^{dim(A(h,w,\chi))} | \lambda \sim \mathcal{N}(0,1) \ \ \forall \lambda \in \Lambda\} \end{aligned} \tag{3.20}$$

where

$$A(h, w, \chi) = \begin{bmatrix} h(w, x_1, t_1) & h(w, x_2, t_2) & \cdots & h(w, x_b, t_b) \end{bmatrix}, \quad (x_i, t_i) \in \chi \ . \tag{3.21}$$

If $h = h_{\mathbb{X}}$, then $\Delta_{\mathbb{X}} R(h, w, \chi)$ is defined as in Balduzzi et al. [1]. Using $h = h_{\mathbb{W}}$ instead results in a metric more closely aligned with $\kappa_{h_{\mathbb{W}}, \mathbb{X}}$ the other metrics introduced later in this section.

As mentioned previously the relative gradient rank is a measure of dissimilarity between gradient samples calculated using elements of a batch. Generally, a matrix constructed using gradient samples as in equation (3.21) will have low rank if the gradient samples (columns) are highly correlated and will have a higher rank if the gradient samples are less similar.

Because the matrix Y in equation (3.20) is randomly sampled its effective rank represents the worst case scenario. Thus a relative effective rank close to 1 means the gradient samples are unstructured, while a relative effective rank closer to 0 means the gradient samples are highly correlated, and thus informative. The relative gradient rank matrix is normalized for gradient magnitudes but does not use any information about the disparity of values within the batch used to calculate it, and thus is a measure of relative variance and not conditioning.

Unfortunately, the gradient rank metric can be prohibitively expensive to calculate. $h_{\mathbb{X}}(w, x, t)$ is usually significantly smaller in dimension than $h_{\mathbb{W}}(w, x, t)$, which may be one reason Balduzzi et al. chose to investigate the conditioning of $h_{\mathbb{X}}(w, x, t)$ instead of $h_{\mathbb{W}}(w, x, t)$ to begin with. Fortunately, the effective rank of the randomly sampled matrix Y is dependent only on the dimensions of A, so we can increase efficiency by calculating the effective rank of a few samples of a given dimension and storing the average for repeated use. Furthermore, the use of random masking can further reduce the cost of calculating $\Delta_{\mathbb{X}} R(h_{\mathbb{W}}, w, \chi)$ as can be seen in figure 3.6 below (also see section 1.6).

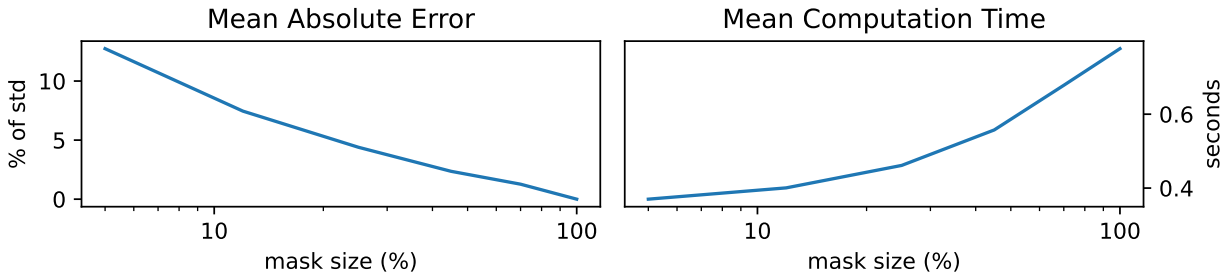


Figure 3.6: Mask Effectiveness for $\Delta_{\mathbb{X}} R(h_{\mathbb{W}}, w, \chi)$

We were able to calculate both $\Delta_{\mathbb{X}} R(h_{\mathbb{X}}, w, \chi)$ and $\Delta_{\mathbb{X}} R(h_{\mathbb{W}}, w, \chi)$ throughout training for several networks of various sizes and architectural styles. The results can be seen in figure 3.7

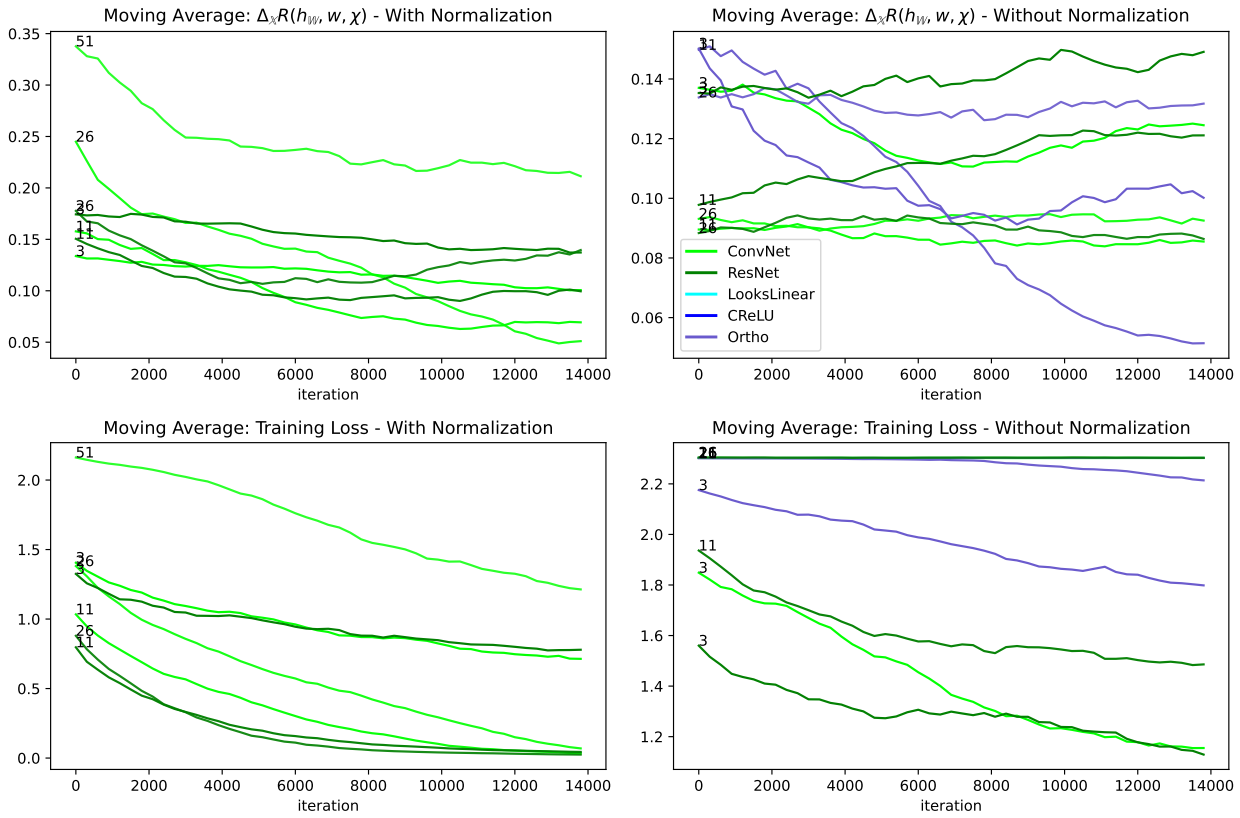


Figure 3.7: The Weight Gradient Rank (see Definition 3.20) was measured throughout training for several different types of architectures. No apparent trends are visible in this view of the collected results, but figure 3.8 contains more compelling trends visible in this metric.
The architectures tested included fully connected networks with orthogonal initialization and the ReLU or CReLU activation function, the Looks-Linear architecture (see Definition 3.9), and convolutional networks with and without residual connections. The label on each line is the number of layers in the architecture.

Notably, $\Delta_{\mathbb{X}} R(h_{\mathbb{X}}, w, \chi)$ tends to decrease throughout training, although the correlation with training speed or final performance does not seem to be strong.

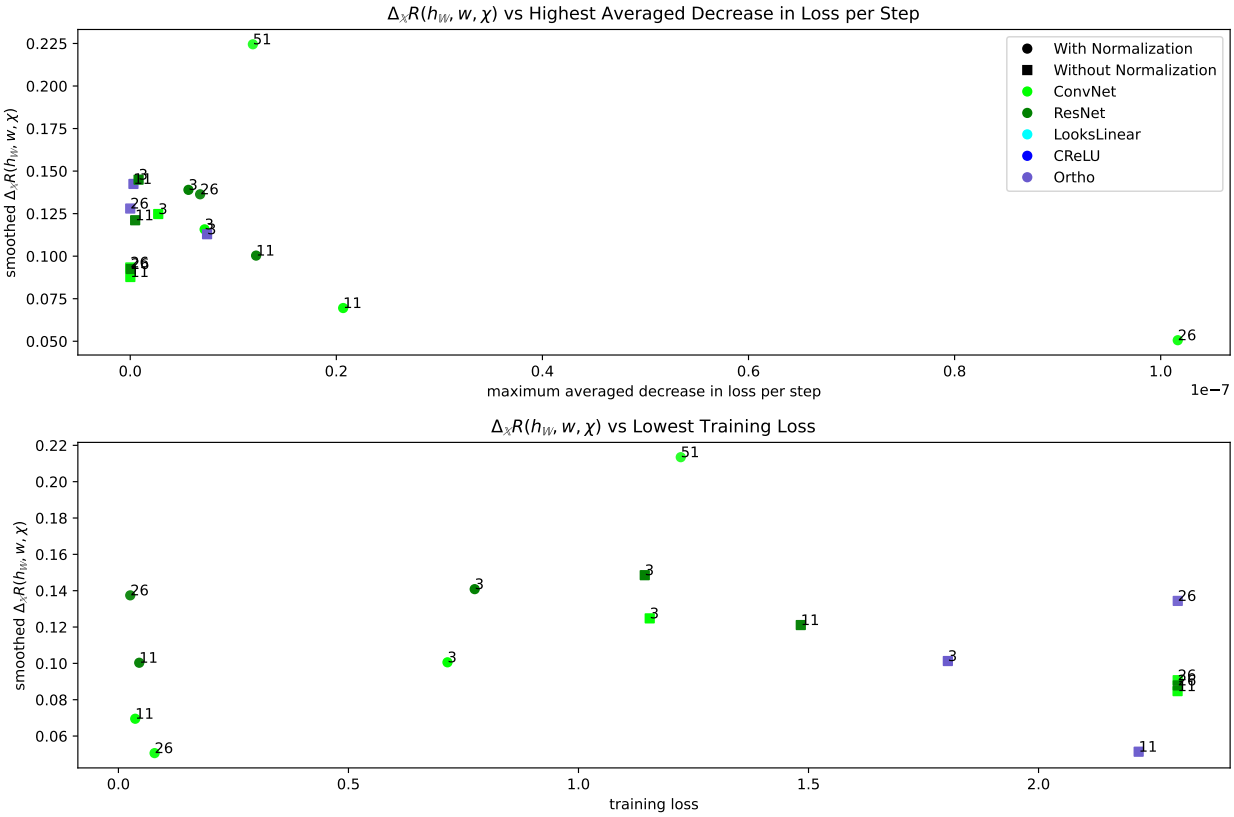


Figure 3.8: The Weight Gradient Rank metric (see Definition 3.20) was measured throughout training for several different types of architectures at the point of steepest descent as well as at the point with the lowest loss (smoothed over a few hundred iterations in each case) and plotted as a point for each network. Excluding the outlier ConvNet26, there is a strong correlation between the fastest rate of descent and the Weight Gradient Rank at that point. A correlation is not evident at the point of lowest loss however.
The architectures tested include fully connected networks with orthogonal initialization and the ReLU or CReLU activation function, the Looks-Linear architecture (see Definition 3.9), and convolutional networks with and without residual connections. The label on each point is the number of layers in the architecture.

$\Delta_{\mathbb{X}} R(h_{\mathbb{W}}, w, \chi)$ seems to have a stronger correlation with training speed and final performance than $\Delta_{\mathbb{X}} R(h_{\mathbb{X}}, w, \chi)$. In particular, networks measuring higher in $\Delta_{\mathbb{X}} R(h_{\mathbb{W}}, w, \chi)$ never learn appreciably at all, although this is also true of some networks that have low $\Delta_{\mathbb{X}} R(h_{\mathbb{W}}, w, \chi)$ values as well.

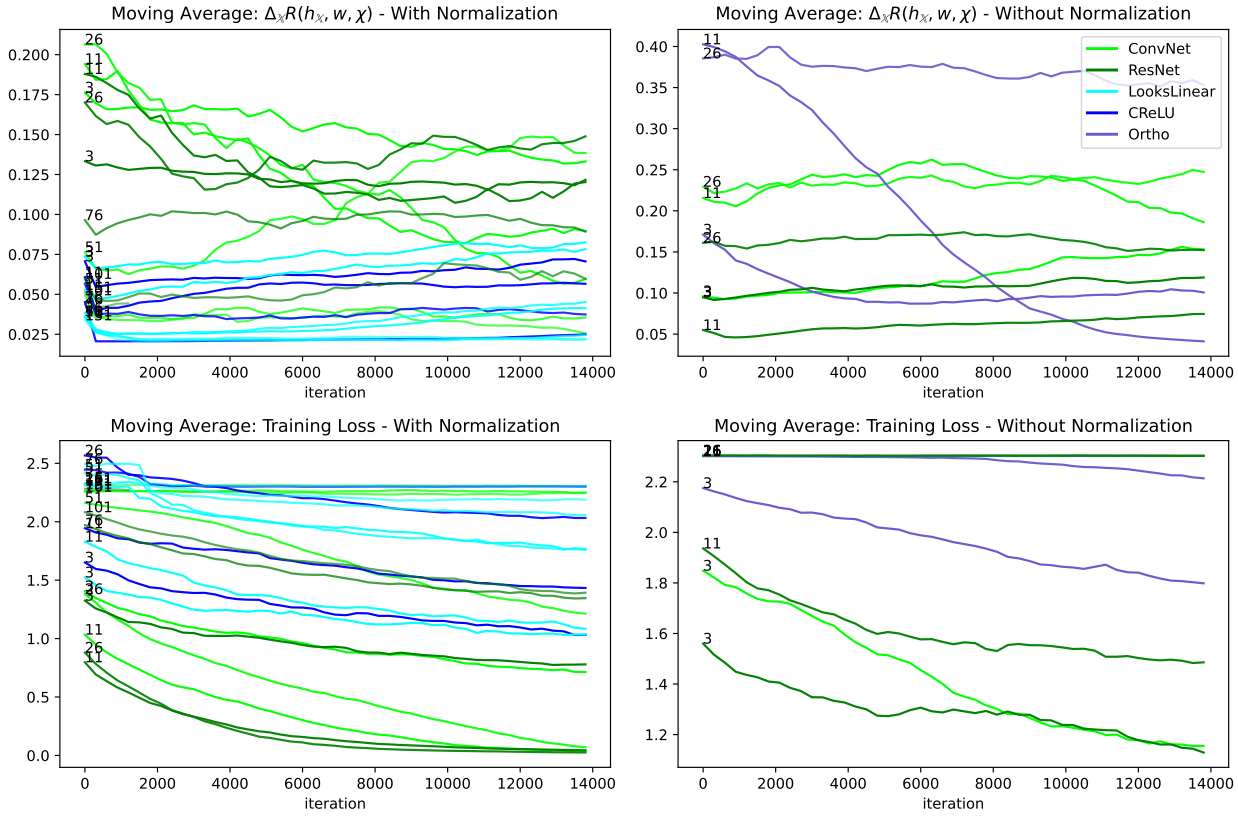


Figure 3.9: The Input Gradient Rank (see Definition 3.20) was measured throughout training for several different types of architectures. Interestingly, some architectures experienced a significant decrease in Input Gradient Rank without converging appreciably.
The architectures tested included fully connected networks with orthogonal initialization and the ReLU or CReLU activation function, the Looks-Linear architecture (see Definition 3.9), and convolutional networks with and without residual connections. The label on each line is the number of layers in the architecture.

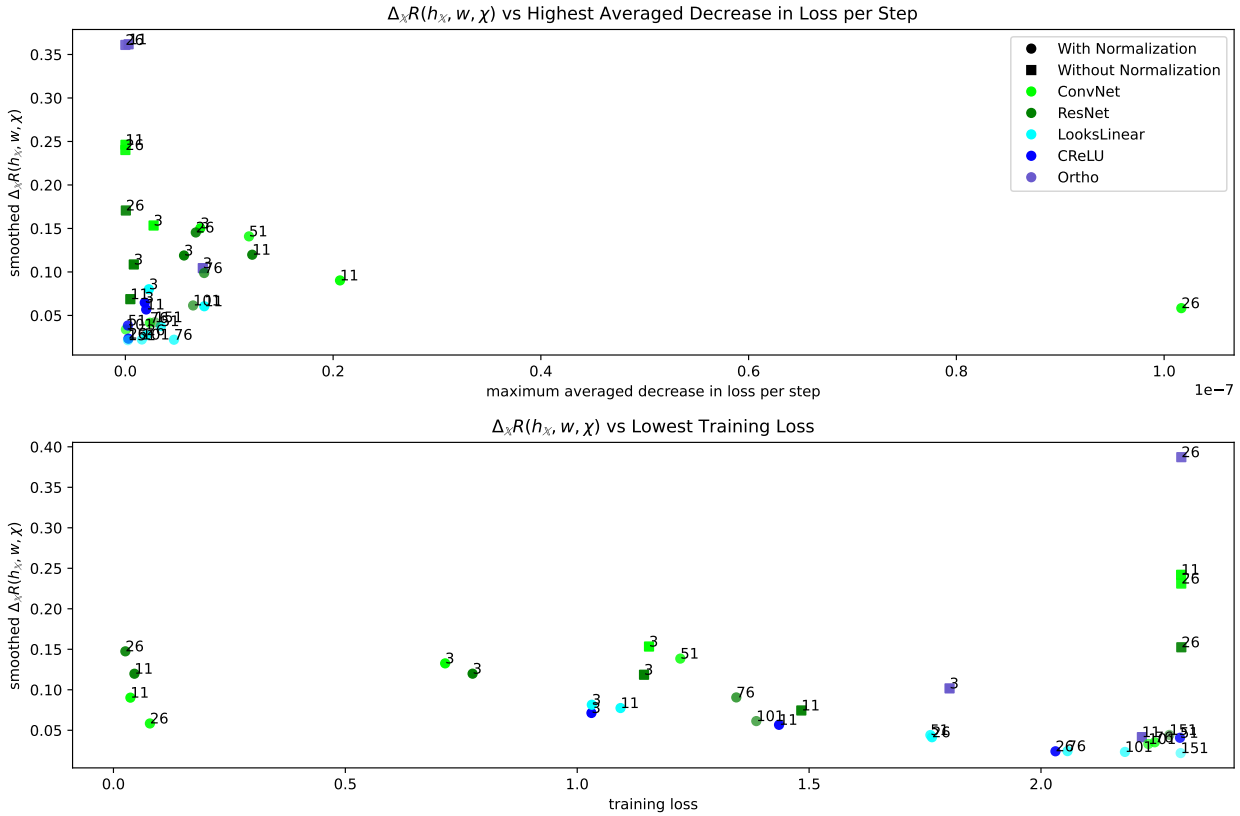


Figure 3.10: The Input Gradient Rank metric (see Definition 3.20) was measured throughout training for several different types of architectures at the point of steepest descent as well as at the point with the lowest loss (smoothed over a few hundred iterations in each case) and plotted as a point for each network. Excluding the outlier ConvNet26, there is a strong correlation between the fastest rate of descent and the Input Gradient Rank at that point. Furthermore, it is apparent that networks with very high Input Gradient Rank often fail to converge.
The architectures tested include fully connected networks with orthogonal initialization and the ReLU or CReLU activation function, the Looks-Linear architecture (see Definition 3.9), and convolutional networks with and without residual connections. The label on each point is the number of layers in the architecture.

3.5.1 Pseudo-Linear Networks and Gradient Whitening.

In order to demonstrate further that gradient whitening significantly impacts SGD in a negative way, Balduzzi et al. carefully construct a network which minimizes the variance in $h_{\mathbb{X}}(w, x, t)$. They do this by carefully initializing a fully connected network in a way that causes the activation functions throughout the network to act like the identity function, thereby removing the discontinuity in every activation function. In order to do this they use an activation function called the *concatenated-ReLU*, or CReLU.

Definition 3.8 (CReLU activation function)**.**

$$CReLU(x) = \begin{bmatrix} ReLU(x) \\ -ReLU(x) \end{bmatrix} . \tag{3.22}$$

The CReLU activation function has significant advantages over ReLU in many applications. For example, in convolutional neural networks, which have a strong tendency to create mirrored filters, using CReLU nearly doubles the effective information density per filter in early layers by automatically supplying the mirrored version of every filter [20].

By initializing the weight matrix of a fully connected layer with a mirrored block structure, $[W\ -W]$, while using a CReLU activation function, Balduzzi et al. create a layer which behaves linearly at initialization, causing $f'(x)$ to be constant for the first step of training.

Definition 3.9 (Looks-Linear Initialization)**.** Let $W \in \mathbb{R}^{m,n}$ be initialized orthogonally. Using a CReLU activation function with a mirrored block structure (each block is W) as shown below results in linear behavior at initialization.

$$[W\ -W] \cdot \begin{bmatrix} ReLU(x) \\ -ReLU(x) \end{bmatrix} = WReLU(x) - WReLU(-x) = Wx\ . \tag{3.23}$$

At initialization, fully-connected layers initialized in this way with CReLU behave perfectly linearly, and have no discontinuities at all. After the first update step affecting each paired set of weights a discontinuity appears, but the discontinuity is significantly smaller than that of non-symmetrically initialized layers and some mirrored pairs persist during training. Figure 3.11 was generated by training a 40-layer fully-connected network. Each layer except the last used the CReLU activation function and was initialized according to Definition 3.9. The pairwise distance was calculated by taking the absolute value of the sum of two weights (since they have opposite signs when paired). We were unable to train a comparable network without symmetric initialization. Every attempt diverged or failed to converge while the symmetrically initialized 40-layer fully-connected network converged

smoothly as long as batch normalization was used, though significantly more slowly than a ResNet with the same number of layers and total parameters did.

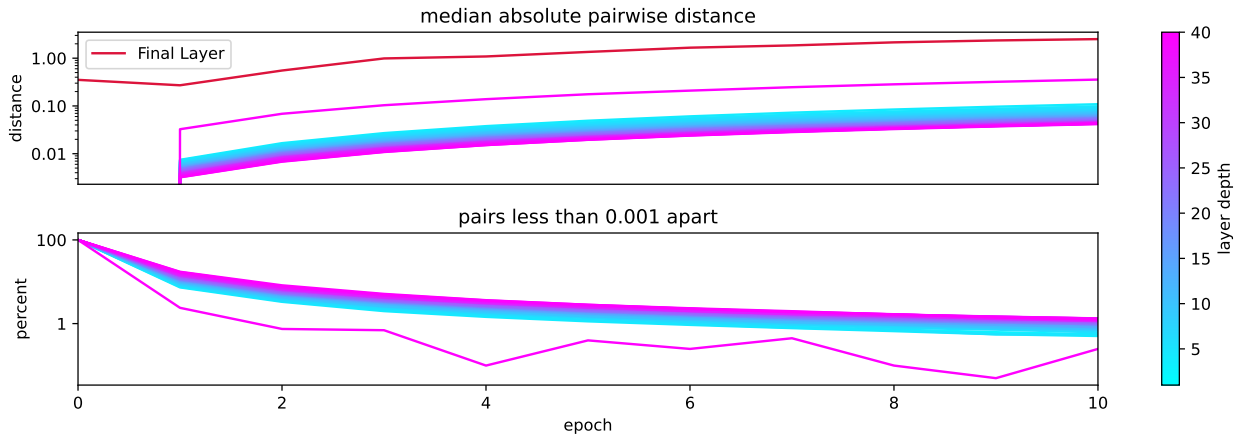


Figure 3.11: Mirrored weights in Looks-Linear Layers persist during training.

Even though fully linear behavior is lost at the first update step, Balduzzi et al. demonstrate that the looks-linear initialization performs at a level with ResNet architectures consistently in architectures up to 200 layers deep, while equivalently sized deep architectures initialized without the mirrored block structure or using ReLU instead of CReLU fail to learn appreciably. We experienced identical results in our own tests.

3.6 Non-Orthogonal Variance and Gradient Confusion

Our previous discussion of variance and conditioning in gradient estimates have operated under the assumption that high variance in the gradient results in poor convergence, which is generally true. However, in their paper "The Impact of Neural Network Overparameterization on Gradient Confusion and Stochastic Gradient Descent" [18], Sankararaman et al. make the more targeted and restricted argument that negative correlation between batch gradient estimates $\overline{h_{\mathbb{W}}(w, \chi_i)}$ interferes with training. In other words, that variance between non-orthogonal portions of different gradient estimates is harmful. In order to study the worst case amount of variance along non-orthogonal directions between batch gradient estimates they define the following bound:

Definition 3.10 (Gradient Confusion Bound η)**.** Let an architecture $g : ((\mathbb{R}^m, ||\cdot||_{\mathbb{W}}), (\mathbb{R}^k, ||\cdot||_{\mathbb{X}})) \to \mathbb{R}^j$ and a loss function $f : (\mathbb{R}^j, T^h) \to \mathbb{R}$ be given. Using the notation given in Definition 1.3, the gradient estimates $\overline{h_{\mathbb{W}}(w, \chi)}$ calculated at a fixed $w \in \mathbb{W}$ using batches from a set of n batches of size b, $\{\chi \in B_b \mid 0 < i \leq n\}$, have gradient confusion bound $\eta \geq 0$ if the pair-wise inner products between gradients satisfy:

$$-\eta \leq \langle \overline{h_{\mathbb{W}}(w, \chi_i)}, \overline{h_{\mathbb{W}}(w, \chi_j)} \rangle, \quad \forall\ \chi_i, \chi_j \in \{\chi \in B_b \mid 0 < i \leq n\},\ i \neq j\ .$$

In order to facilitate efficient comparison of this bound between networks of different sizes, we define a normalized estimate of this bound over a fixed set of n sampled pairs of batches.

Definition 3.11 (Gradient Confusion Bound Estimate $\Delta_{\mathbb{W}} C_n(w)$)**.** Let an architecture $g : ((\mathbb{R}^m, ||\cdot||_{\mathbb{W}}), (\mathbb{R}^k, ||\cdot||_{\mathbb{X}})) \to \mathbb{R}^j$ and a loss function $f : (\mathbb{R}^j, T^h) \to \mathbb{R}$ be given. Using the notation given in Definition 1.3, we define the gradient confusion bound estimate over a set of n pairs of batches of size b as follows:

$$\Delta_{\mathbb{X}} C_n(w) := -\min\left(\min_{(\chi_i, \chi_j) \in \zeta_n} \left(\frac{\langle \overline{h_{\mathbb{W}}(w, \chi_i)}, \overline{h_{\mathbb{W}}(w, \chi_j)} \rangle_{\mathbb{W}}}{||\overline{h_{\mathbb{W}}(w, \chi_i)}||_{\mathbb{W}} \cdot ||\overline{h_{\mathbb{W}}(w, \chi_j)}||_{\mathbb{W}}} \right),\ 0 \right)\ , \tag{3.24}$$

$$\text{where}\ \ \zeta_n = \{(\chi_i, \chi_j)_k \mid \chi_i, \chi_j \in B_b(\mathbb{X}),\ i \neq j,\ 0 < k < n\}\ .$$

We note that Sankararaman et al. also used a sampled set of n pairs of batches to estimate η in practice instead of using every possible pairing from a given set of batches. Furthermore, the normalization is performed as suggested at the end of their paper, making $\Delta_{\mathbb{W}} C_n(w)$ equivalent to the most negative cosine similarity over the sampled pairs of batches.

We were able to measure gradient confusion throughout training for a significant variety of network depths and architectural styles. The results can be seen in figures 3.12 and 3.13 below.

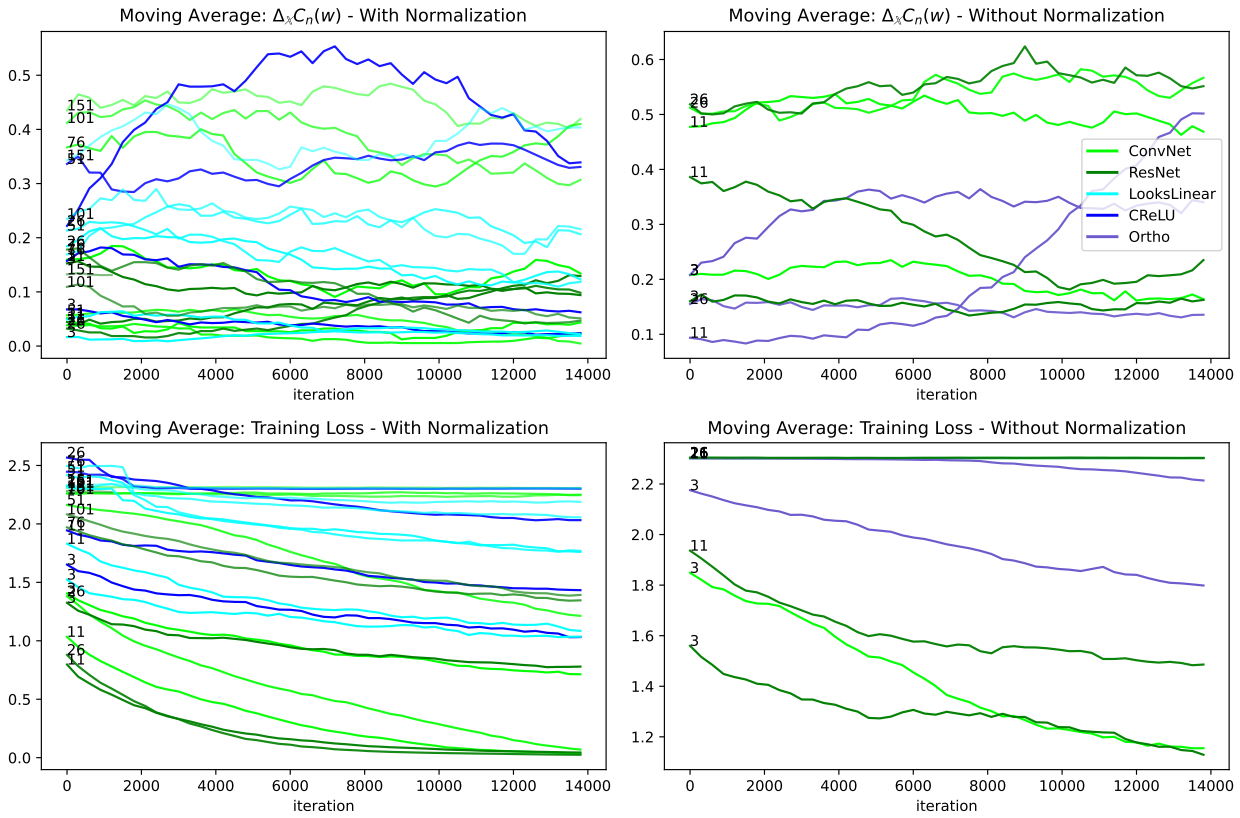


Figure 3.12: The Gradient Confusion metric (see Definition 3.24) was measured throughout training for several different types of architectures. It is difficult to observe compelling trends in this figure about the Gradient Confusion Metric (figure 3.13 instead), however, it is interesting to note that the Gradient Confusion can sometimes change significantly for a network over many epochs of training that fail to decrease the loss appreciably (see Ortho11, which begins to converge only near the end of the testing window but for which Gradient Confusion increases consistently throughout training).
The architectures tested included fully connected networks with orthogonal initialization and the ReLU or CReLU activation function, the Looks-Linear architecture (see Definition 3.9), and convolutional networks with and without residual connections. The label on each line is the number of layers in the architecture.

Notably, there is a strongly pronounced correlation between gradient confusion and training ability. In figure 3.13 below it can be seen that final performance and speed of training are both strongly correlated with gradient confusion measurements, with low gradient confusion being better of course.

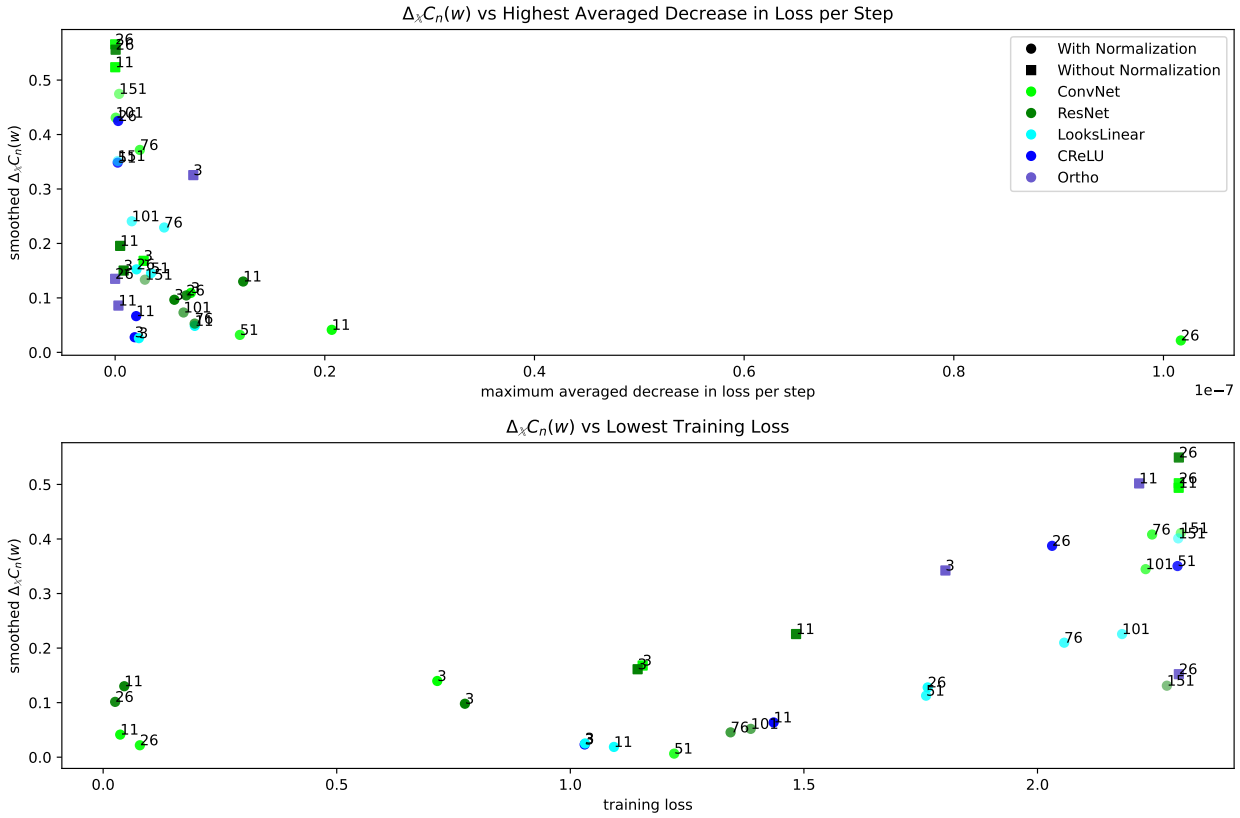


Figure 3.13: The Gradient Confusion metric (see Definition 3.24) was measured throughout training for several different types of architectures at the point of steepest descent as well as at the point with the lowest loss (smoothed over a few hundred iterations in each case) and plotted as a point for each network. There is a clear correlation between the fastest rate of descent and the Gradient Confusion at the point of steepest descent. There is also a strong correlation between lower Gradient Confusion and improved final performance.
The architectures tested include fully connected networks with orthogonal initialization and the ReLU or CReLU activation function, the Looks-Linear architecture (see Definition 3.9), and convolutional networks with and without residual connections. The label on each point is the number of layers in the architecture.

Theories and Results from Sankararaman et al. There are a significant number of theories and interesting implications in this paper, but we will restrict ourselves to the most relevant, namely:

(i) Extremely low gradient confusion measurements are correlated with fast convergence and a lower noise floor at convergence (i.e. better final performance).

(ii) They demonstrate that increasing width in an architecture of fixed depth reduces gradient confusion and speeds network training.

(iii) They demonstrate that increasing depth in a narrow network increases gradient confusion and makes the network difficult to train.

(iv) They find that adding skip-connections or batch normalization both individually improve training and reduce gradient confusion. However, they note specifically that using both together enabled the training of much deeper networks than using either alone, and that gradient confusion measurements tend to be very low even in very deep networks in this case.

(v) When gradient magnitude and variance are bounded reducing the learning rate is guaranteed to reduce the variance between gradient estimates taken before and after each step.

(vi) Finally, they prove that linear networks (i.e. fully connected networks without activation functions) with orthogonal initialization are independent of gradient confusion at initialization.

These results and conclusions agree very well with findings from other papers we have cited, and we expect that we can validate these findings in our own tests. Item (iv) above was particularly striking to us, although not surprising. Linear networks are generally not used in practice because they cannot express non-linear behavior. Without activation functions between layers the network can be collapsed to a single matrix multiplication with no loss of information, hence the independence of depth if each layer is initialized orthogonally. However, the Looks-Linear Initialization discussed in the previous section (see Definition 3.9) is a peculiar network configuration that has the potential to learn non-linear behavior, but is linear at initialization. Even more interesting is that it performs astoundingly well for a fully-connected network. We were able to train a 200-layer fully-connected network using CReLU and the Looks-Linear initialization to similar performance as a 200-layer ResNet. Unfortunately, however, we did not observe a striking difference in our measurements of the Looks-Linear architecture compared to other linear networks. Because of a package

incompatibility we were unable to measure our full set of metrics on any networks containing the CReLU activation function, so we intend to investigate the Looks-Linear initialization further in the future.

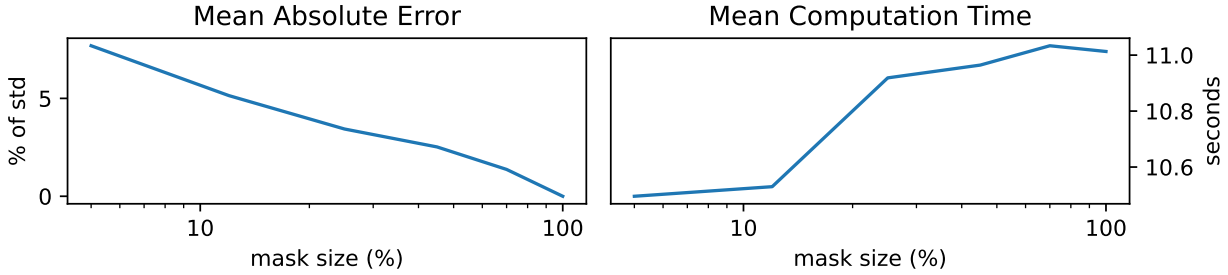


Figure 3.14: Mask Effectiveness for the Gradient Confusion Metric

It should be noted that while Sankararaman et al. established concrete and meaningful results using minimum cosine similarity, they were not the only group to use it as a gradient metric in deep learning. Minimum cosine similarity was used separately by [27]. Since Sankararaman et al. published this paper however, the Gradient Confusion metric as defined by Sankararaman et al. has been used or referenced in several other papers: [3, 6, 11, 23]. Of these papers, we are most interested in the results of Guilin et al. [11], who successfully used a normalized form of the Gradient Confusion metric as the optimization parameter in an architecture search algorithm, which is one of our own objectives.

3.7 Gradient Diversity

While most of the papers we have found that are related to our topics of study are interested in minimizing the variance between gradients estimated with different inputs, the authors of [29] focus on the detriments caused by too little variance between gradients for different inputs. Yin et al. claim that too much similarity can be detrimental to stability in learning near the end of training, and can cause significant loss in the ability of a network to generalize. They use a metric they name Gradient Diversity to measure the variance between gradients calculated with various sizes of mini-batches. Notably, this metric is equally well, if not better, suited to measuring high variance between batch gradient estimates. We will formally

define gradient diversity as it is used by Yin et al. and measure it in our own tests with an interest in both the high and low variance cases.

Definition 3.12 (Gradient Diversity)**.** Let an architecture $g : ((\mathbb{R}^m, ||\cdot||_{\mathbb{W}}), (\mathbb{R}^k, ||\cdot||_{\mathbb{X}})) \to \mathbb{R}^j$ and a loss function $f : (\mathbb{R}^j, T^h) \to \mathbb{R}$ be given. Using the notation given in Definition 1.3, we define the *Gradient Diversity* $\Delta_{\mathbb{X}} D_n(w)$ metric using a set of n batches $\chi_i \in B_b$ as follows:

$$\Delta_{\mathbb{X}} D_n(w) := \frac{\sum_{i=1}^{n} \left\| \overline{h_{\mathbb{W}}(w, \chi_i)} \right\|_2^2}{\left\| \sum_{i=1}^{n} \overline{h_{\mathbb{W}}(w, \chi_i)} \right\|_2^2} = \frac{\sum_{i=1}^{n} \left\| \overline{h_{\mathbb{W}}(w, \chi_i)} \right\|_2^2}{\sum_{i=1}^{n} \left\| \overline{h_{\mathbb{W}}(w, \chi_i)} \right\|_2^2 + \sum_{i \neq j} \langle \overline{h_{\mathbb{W}}(w, \chi_i)}, \overline{h_{\mathbb{W}}(w, \chi_j)} \rangle_{\mathbb{W}}} \tag{3.25}$$

with the final equality following via the linearity and symmetry conditions of the inner product over the real space $\mathbb{R}^m$.

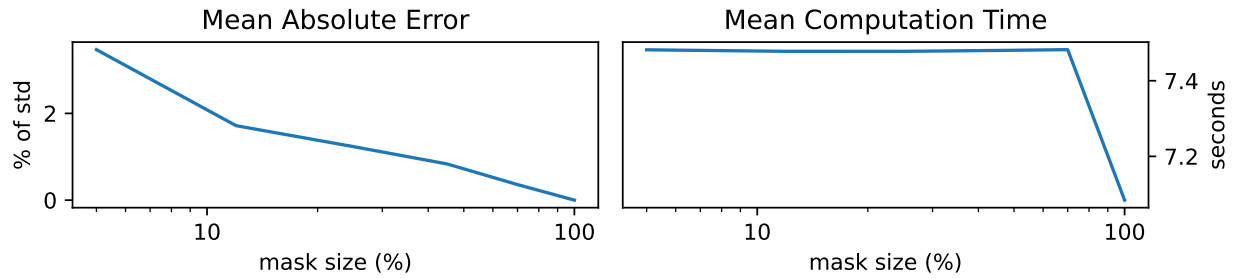


Figure 3.15: Because Gradient Diversity requires several backward calls, and the vector two norm is so cheap to calculate, it is significantly cheaper temporally to use the full gradient vector than to mask it. Gradient Diversity also requires very little additional memory to calculate.

Because the only interaction between gradient estimates for different batches in the above formula is in an inner product, the magnitudes calculated serve only in a normalizing capacity. In that sense gradient diversity measures much the same thing as the gradient confusion. Specifically, gradient diversity is concerned with the net dissimilarity between batch gradient estimates while gradient confusion measures the worst case dissimilarity. The gradient diversity metric has a large advantage in numerical implementation over gradient confusion or the condition numbers however. Namely, it doesn't require temporary storage of every gradient sample used and doesn't actually require the calculation of pairwise inner products. Instead samples of $\overline{h_{\mathbb{W}}(w, \chi)}$ and $||\overline{h_{\mathbb{W}}(w, \chi)}||_2^2$ can be summed as they are drawn. Note that

the gradient diversity does have a dependence on the sample size n, thus the same sample size should be used when comparing any two given networks.

We were able to measure gradient diversity throughout training for a variety of network depths and architectural styles. The results can be seen in figures 3.16 and 3.17.

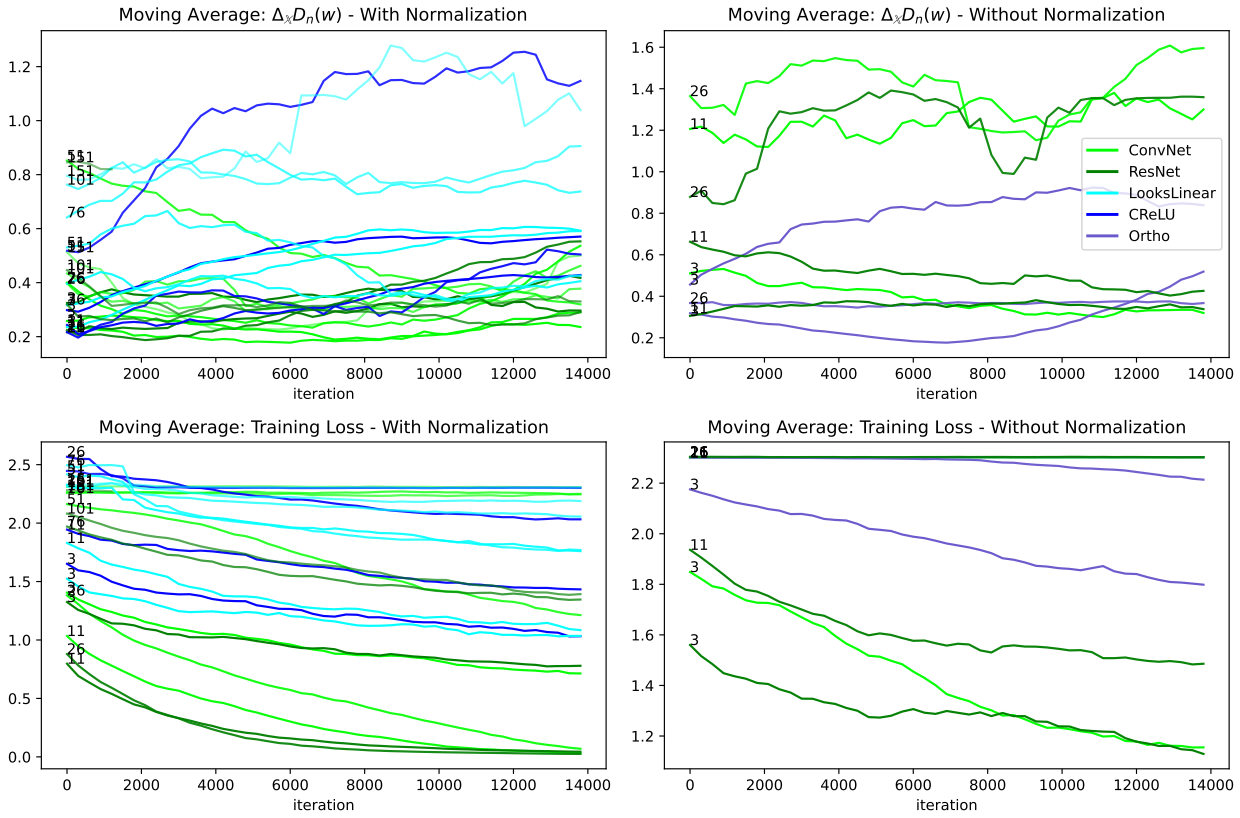


Figure 3.16: The Gradient Diversity metric (see Definition 3.25) was measured throughout training for several different types of architectures. It is difficult to observe compelling trends in this figure (see figure 3.17 instead). In general however, it seems that gradient diversity tends to increase slowly throughout training when networks converge successfully.
The architectures tested included fully connected networks with orthogonal initialization and the ReLU or CReLU activation function, the Looks-Linear architecture (see Definition 3.9), and convolutional networks with and without residual connections. The label on each line is the number of layers in the architecture.

Similar to the Gradient Confusion metric, there is a pronounced correlation between gradient diversity and training ability. In figure 3.17 it can be seen that final performance and speed of training are both correlated with low gradient diversity.

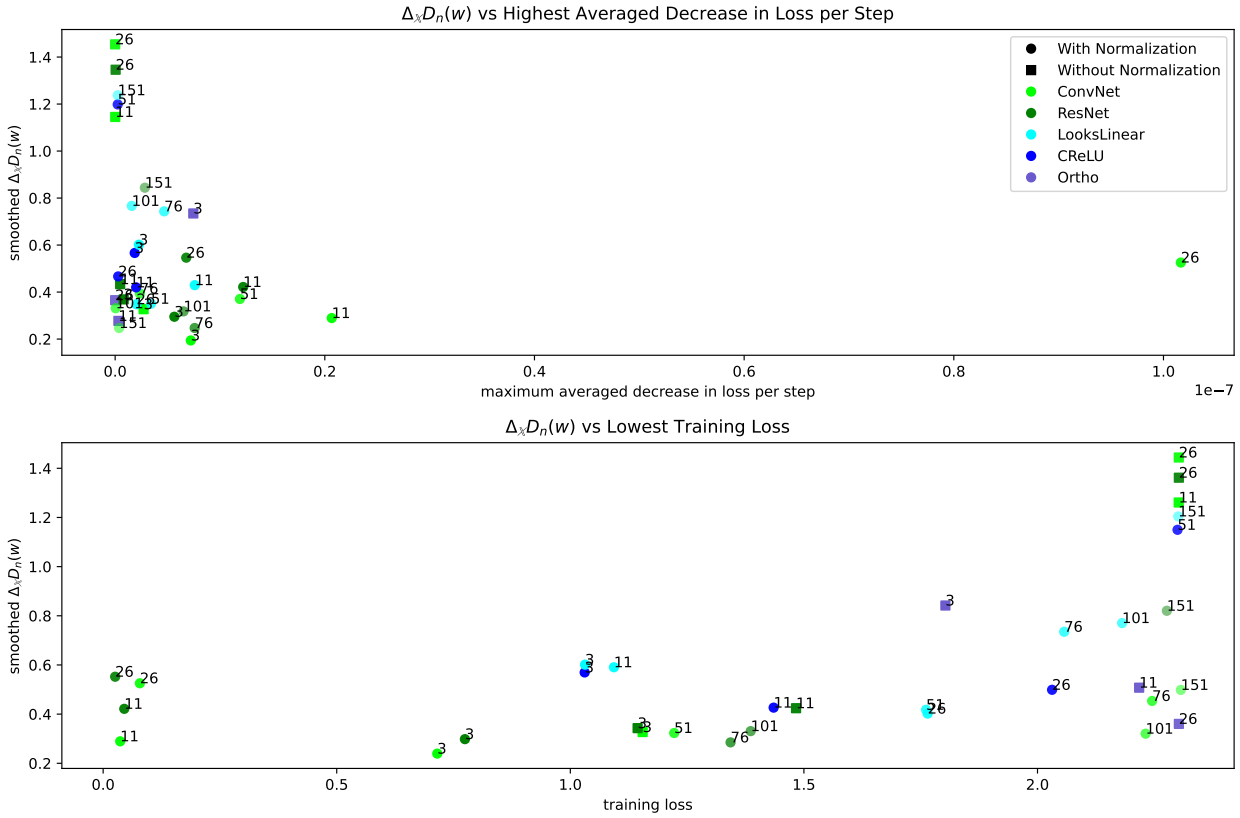


Figure 3.17: The Gradient Diversity metric (see Definition 3.25) was measured throughout training for several different types of architectures at the point of steepest descent as well as at the point with the lowest loss (smoothed over a few hundred iterations in each case) and plotted as a point for each network. There is a clear correlation between the fastest rate of descent and the Gradient Diversity at that point. Furthermore, there is a strong correlation between lower Gradient Diversity and improved final performance as well.
The architectures tested include fully connected networks with orthogonal initialization and the ReLU or CReLU activation function, the Looks-Linear architecture (see Definition 3.9), and convolutional networks with and without residual connections. The label on each point is the number of layers in the architecture.

A few other papers have referenced and used the gradient diversity, including [10, 27] on adversarial robustness and transfer.

3.7.1 Is Variance Beneficial. It is clear that extremely low variance between gradient estimates can slow learning significantly. This has been demonstrated in several papers, including Ghorbani et al. [5], Neelakantan et al. [13], and the two papers just cited on adversarial robustness. In figure 3.17 it appears as though low measurements in the gradient diversity metric strictly benefit a network during training, in particular near the end of train-

ing. This was also the case for the gradient confusion metric. However, low measurements in the metric Gradient Coherence we introduce in the following section is correlated with weaker performance in some types of networks (see figure 3.20).

3.8 Gradient Coherence

The past few years has seen a significant interest in a particular phenomenon. Specifically, why do networks trained on real datasets (namely ones where training labels are assigned via some non-random process) generalize so much better than networks trained with randomly generated labels when similar accuracy is obtained on the training set. The intuitive answer is that networks trained with meaningful labels learn generalizeable trends in the data, and networks trained with random labels are forced to memorize inputs because trends in the data are dissociated from the training label.

The paper "Coherent Gradients: An Approach to Understanding Generalization in Gradient Descent-Based Optimization" [2] by Satrajit Chatterjee at Google AI proposed a straightforward theory along these lines. Though the suggested ideas were present to an extent in the literature prior to this time, Chatterjee puts it all together with the name the Coherent Gradients hypothesis, included here:

> 1. Gradients are *coherent*, i.e, similar examples (or parts of examples) have similar gradients (or similar components of gradients) and dissimilar examples have dissimilar gradients.
>
> 2. Since the overall gradient is the sum of the per-example gradients, it is stronger in directions where the per-example gradients are similar and reinforce each other and weaker in other directions where they are different and do not add up.
>
> 3. Since network parameters are updated proportionally to gradients, they change faster in the direction of stronger gradients.

4. Thus the changes to the network during training are biased towards those that simultaneously benefit many examples instead of a few (or one example).

They note that previous papers (published at ICLR 2020) focus their analysis on converging to generalizeable points in weight space, but Chatterjee argues that algorithmic stability, and generalizeability at all points during training, is more likely to yield meaningful results. This theorem and idea align extremely well with our analysis on gradient conditioning with respect to the input space, though the initial motivation is different.

The subsequent paper, "Making Coherence Out of Nothing at All: Measuring the Evolution of Gradient Alignment" [3] by Chatterjee and Piotr Zielinski, also at Google AI, introduces a metric to measure this, defined as follows:

Definition 3.13 (Gradient Coherence)**.** Let an architecture $g : ((\mathbb{R}^m, ||\cdot||_{\mathbb{W}}), (\mathbb{R}^k, ||\cdot||_{\mathbb{X}})) \to \mathbb{R}^j$ and a loss function $f : (\mathbb{R}^j, T^h) \to \mathbb{R}$ be given. Using the notation given in Definition 1.3, we define the *Gradient Coherence* as follows:

$$\Delta_{\mathbb{X}} H(w) := \frac{\mathop{\mathbb{E}}_{(x_i,t_i)\sim\mathbb{X}\times\mathbb{T}^h} [\langle h_{\mathbb{W}}(w), h_{\mathbb{W}}(w, x_i, t_i)\rangle_{\mathbb{W}}]}{\mathop{\mathbb{E}}_{(x_i,t_i)\sim\mathbb{X}\times\mathbb{T}^h} [\langle h_{\mathbb{W}}(w, x_i, t_i), h_{\mathbb{W}}(w, x_i, t_i)\rangle_{\mathbb{W}}]} .$$

For a sample size b this can be calculated as

$$\Delta_{\mathbb{X}} H_b(w) := \frac{\frac{1}{b}\sum_{(x_i,t_i)\in\chi} \langle \overline{h_{\mathbb{W}}(w,\chi)}, h_{\mathbb{W}}(w, x_i, t_i)\rangle_{\mathbb{W}}}{\frac{1}{b}\sum_{(x_i,t_i)\in\chi} \langle h_{\mathbb{W}}(w, x_i, t_i), h_{\mathbb{W}}(w, x_i, t_i)\rangle_{\mathbb{W}}} = \frac{\sum_{(x_i,t_i)\in\chi} \langle \overline{h_{\mathbb{W}}(w,\chi)}, h_{\mathbb{W}}(w, x_i, t_i)\rangle_{\mathbb{W}}}{\sum_{(x_i,t_i)\in\chi} ||h_{\mathbb{W}}(w, x_i, t_i)||^2_{\mathbb{W}}} \tag{3.26}$$

where $\chi \in B_b$.

Notably, this is extremely similar to both the gradient confusion and gradient diversity metrics, and this is acknowledged at introduction. However, Chatterjee et al. make a very particular choice of normalization constant, which allows them to demonstrate that their metric, $(\Delta_{\mathbb{X}} H(w))$,

"is the change in the overall loss due to a small gradient step as a fraction of the maximum possible change in loss if each component of the loss could be optimized

independently."

Furthermore, they show that $\Delta_{\mathbb{X}} H_b(w)$ is the average fraction of elements in $\chi \in B_b$ that a given $h_{\mathbb{W}}(w, x_i, t_i)$ decreases the loss for, itself included. This is more easily interpretable than the gradient confusion or gradient diversity metrics, and like the other two, is scale invariant.

The gradient coherence metric has a further advantage over the gradient confusion, namely that it can be calculated in $O(b)$ time without compromising the search space where gradient confusion relies on sampling a subset of possible pairs over the targeted sample space to avoid a $O(b^2)$ calculation cost.

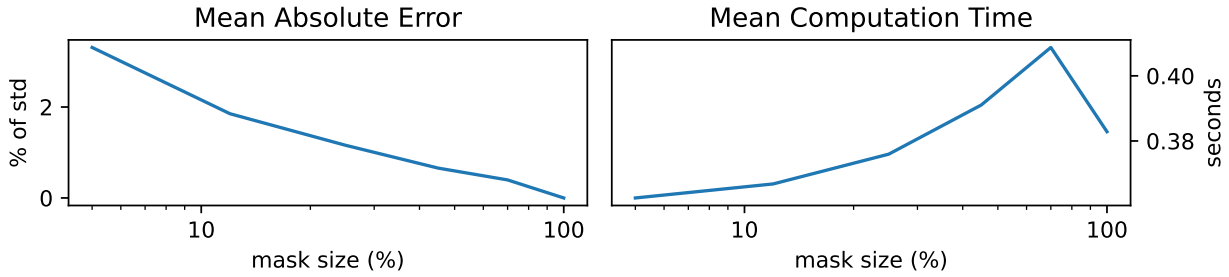


Figure 3.18: Mask Effectiveness for Gradient Coherence Metric

If the batch-size b is large, $\Delta_{\mathbb{X}} H_b(w)$ appears to be an extremely suitable measure on gradient variance induced by differences between inputs within a batch, and is likely more consistent than the gradient confusion because it relies on calculating differences between the batch gradient estimate and each individual gradient estimate, rather than pairwise differences. In this way it is more closely related to the defined condition number, though it also does not normalize for the size of difference between inputs and is therefore more dependent on the known training set and given batch than the condition number is.

We measured gradient coherence throughout training for a variety of network depths and architectural styles. The results can be seen in figures 3.19 and 3.20 below.

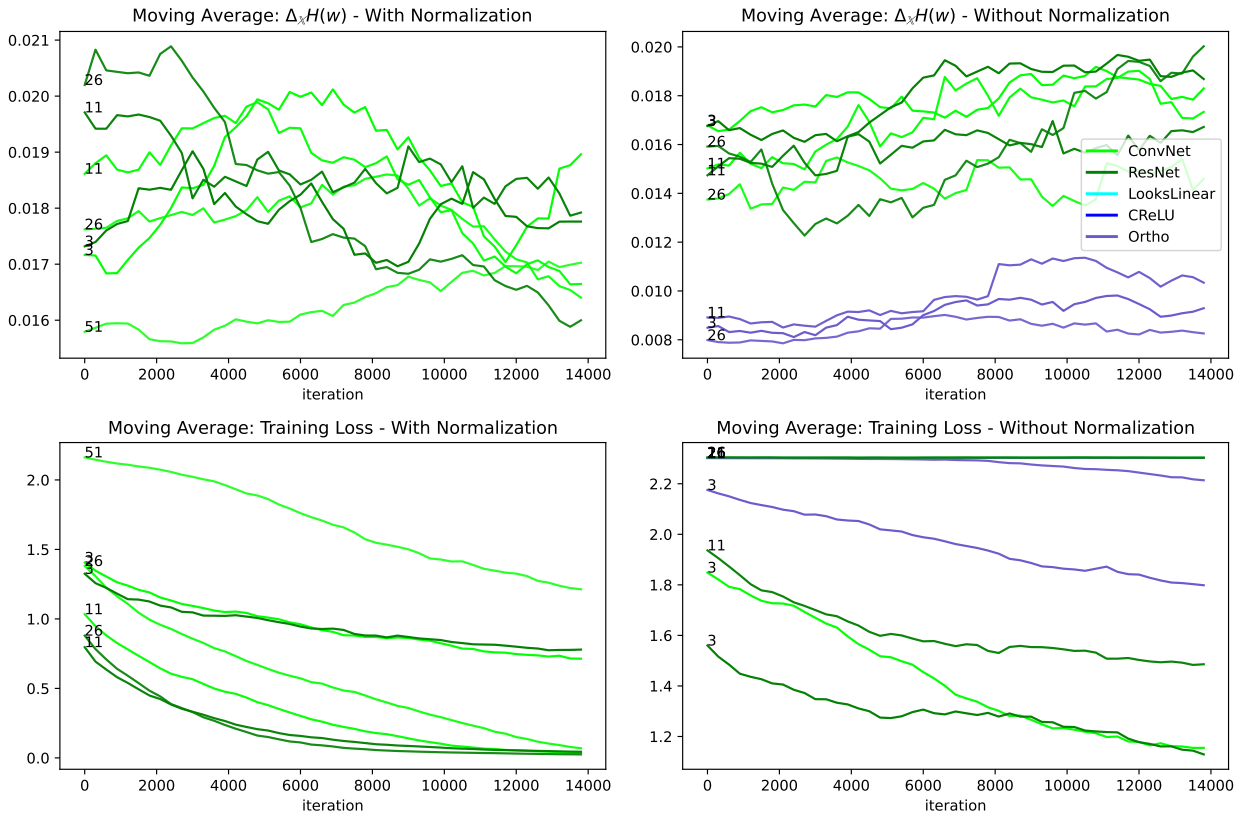


Figure 3.19: The Gradient Coherence metric (see Definition 3.13) was measured throughout training for several different types of architectures. There is a clear division in this metric between convolutional and fully-connected networks that doesn't seem to be associated with convergence. The architectures tested included fully connected networks with orthogonal initialization and the ReLU or CReLU activation function, the Looks-Linear architecture (see Definition 3.9), and convolutional networks with and without residual connections. The label on each line is the number of layers in the architecture.

Interestingly there is a pronounced difference in gradient coherence measurements between fully connected networks and convolutional ones, with fully connected networks measuring uniformly lower in coherence. Otherwise, gradient coherence does not seem strongly predictive of training performance between convolutional networks (see figure 3.20 below). It does appear from our limited set of tests that extreme values of coherence in either direction in convolutional networks is an indicator of poor performance.

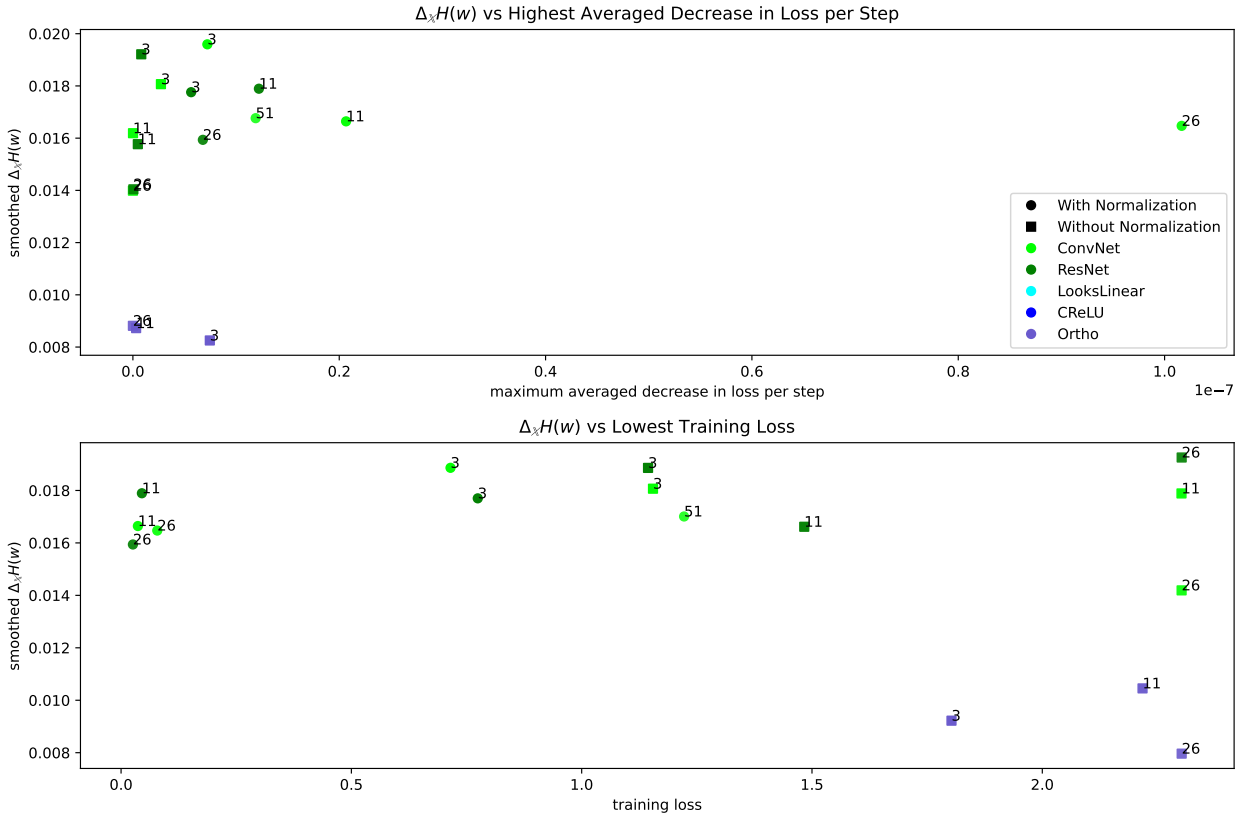


Figure 3.20: The Gradient Coherence metric (see Definition 3.13) was measured throughout training for several different types of architectures at the point of steepest descent as well as at the point with the lowest loss (smoothed over a few hundred iterations in each case) and plotted as a point for each network. Interestingly, there is a strong separation between fully-connected architectures and convolutional architectures in both graphs, but no other trends are apparent.
The architectures tested include fully connected networks with orthogonal initialization and the ReLU or CReLU activation function, the Looks-Linear architecture (see Definition 3.9), and convolutional networks with and without residual connections. The label on each point is the number of layers in the architecture.

Chapter 4. Remaining Objectives and Future Work

Recall from chapter 1 that our stated objectives are the following:

1. Summarize previous findings related to gradient step-consistency and batch-dissonance and place them in the uniform context of gradient conditioning.

2. Demonstrate efficient methods by which gradient step-consistency and batch-dissonance

can be measured independently.

3. Investigate the impact gradient step-consistency and batch-dissonance each have on the stability and speed of SGD.

4. If possible, demonstrate that variance and instability from each source can be reduced or mitigated independently and in a targeted way using architectural or hyperparameter adjustments.

Our intent with objective 1 was accomplished in chapters 2 and 3. We have also accomplished what he had hoped to accomplish for objectives 2 and 3 with regards to gradient batch-dissonance. However, we were unable to find the the number and variety of metrics on gradient step-consistency that we were able to find for gradient batch-dissonance. This is also an obstacle to accomplishing objective 4. We will attempt to address this lack in the following section. Thereafter we will consider what is required to accomplish objective 4 in our future work section 4.2.

4.1 Adapting Batch-Dissonance Metrics to Measure Gradient Step-Consistency between Steps

Given the relative lack of viable metrics to measure gradient step-consistency across optimization steps in chapter 2 compared to chapter 3 we will adapt a few of the metrics from the later chapter to measure relative variance between samples of $\overline{h_{\mathbb{W}}(w, \chi)}$ with varying w instead of $h_{\mathbb{W}}(w, x, t)$ with varying (x, t). Consider the following definition.

Definition 4.1 (**Batch-Dissonance metrics modified to measure Step-Consistency**)**.** Let an architecture $g : ((\mathbb{R}^m, ||\cdot||_{\mathbb{W}}), (\mathbb{R}^k, ||\cdot||_{\mathbb{X}})) \to \mathbb{R}^j$ and a loss function $f : (\mathbb{R}^j, T^h) \to \mathbb{R}$ be given. Using the notation given in Definition 1.3, we define the minimum weight-gradient cosine similarity, weight-gradient diversity, and weight-gradient coherence for a sample size $n \in \mathbb{N}$ as follows:

$$\Delta_{\mathbb{W}} C_n^*(w,\chi) := \min_{0<i\leq n} \left(\frac{\langle \overline{h_{\mathbb{W}}(w+\omega_i,\chi)}, \overline{h_{\mathbb{W}}(w,\chi)} \rangle_{\mathbb{W}}}{||\overline{h_{\mathbb{W}}(w+\omega_i,\chi)}||_{\mathbb{W}} \cdot ||\overline{h_{\mathbb{W}}(w,\chi)}||_{\mathbb{W}}} \right) , \tag{4.1}$$

$$\Delta_{\mathbb{W}} D_n(w,\chi) := \frac{\sum_{i=1}^n \left\| \overline{h_{\mathbb{W}}(w+\omega_i,\chi)} \right\|_2^2}{\left\| \sum_{i=1}^n \overline{h_{\mathbb{W}}(w+\omega_i,\chi)} \right\|_2^2} , \tag{4.2}$$

and

$$\Delta_{\mathbb{W}} H_n(w,\chi) := \frac{\sum_{i=1}^n \langle \overline{h_{\mathbb{W}}(w,\chi)}, \overline{h_{\mathbb{W}}(w+\omega_i,\chi)} \rangle_{\mathbb{W}}}{\sum_{i=1}^n ||\overline{h_{\mathbb{W}}(w+\omega_i,\chi)}||_{\mathbb{W}}^2} \tag{4.3}$$

where $w \in \mathbb{W}$, $\chi \in B_b$, and $\omega_i \in \{\omega \in \mathbb{W} \mid 0 < ||\omega|| < \epsilon\}$ for some $\epsilon > 0$.

More significant adaptations were required for the modified Gradient Confusion metric ($\Delta_{\mathbb{W}} C_n^*(w,\chi)$ in equation (4.1)) than for the other two. Gradient Confusion was designed to consider only gradients that are partially in opposite directions, but this will very rarely be the case with a small perturbation as defined here. Because of that it was necessary to include non-negative cosine similarity values (by removing the outer minimum), otherwise the metric would be uniformly zero for almost all network states.

Because the metrics these were derived from were all designed to operate on vectors that differ considerably more than these gradient estimates will after a small perturbation, the expected variance between samples is quite small. We can maximize the expressivity of each metric in a consistent way by using the targeted sampling technique from Definition 2.6 or Definition 2.7. In other words, these metrics will be the most representative of behavior during SGD if gradient samples are used as perturbation directions. If they are calculated at the same time as the gradient conditioning with respect to the weight space the same samples can be used.

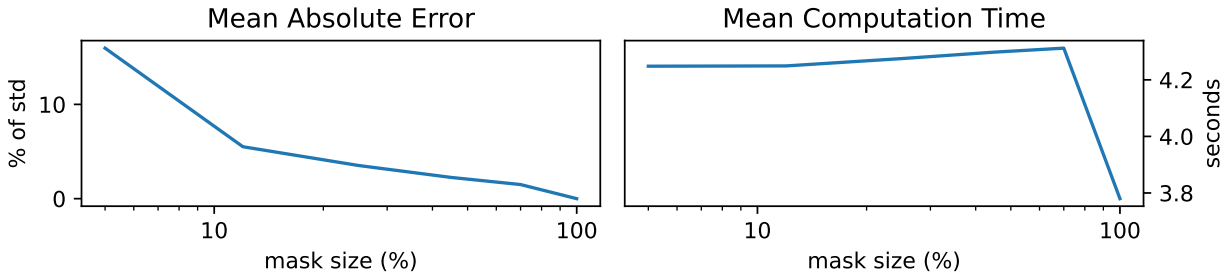


Figure 4.1: Mask Effectiveness for $\Delta_{\mathbb{W}} D_n(w, \chi)$

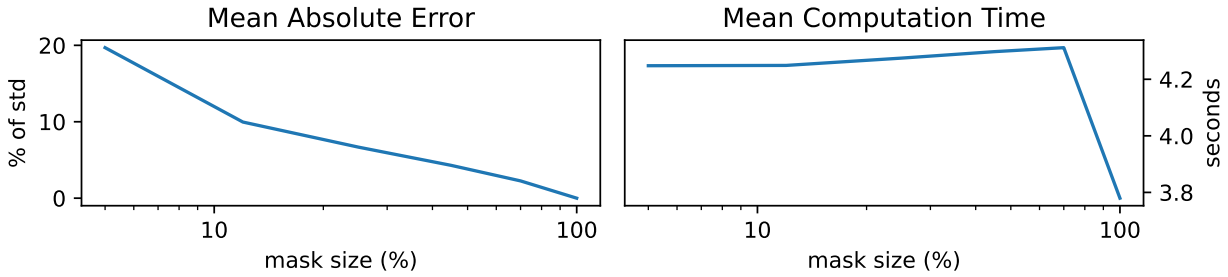


Figure 4.2: Mask Effectiveness for $\Delta_{\mathbb{W}} H_n(w, \chi)$

No figure on the mask effectiveness of the minimum weight-gradient cosine similarity is available because the modifications noted at the bottom of Definition 4.1 were made after the mask-accuracy test was completed and the test takes a significant amount of time to run.

The effect of the targeted sampling methods from section 2.3 on the samples used in the minimum weight-gradient cosine similarity can be seen in figure 4.3.

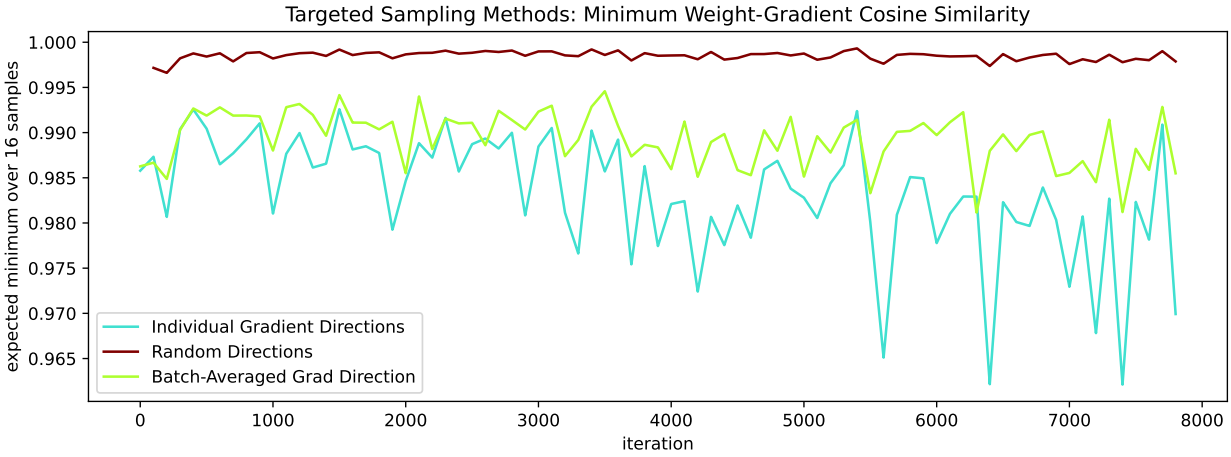


Figure 4.3: Targeted Sampling Methods: Minimum Weight-Gradient Cosine Similarity. During the training of an 11 layer convolutional network (with normalization but without skip connections), 64 input perturbation directions were generated via the sampling methods described in section 2.3. Bootstrapping was used to estimate an expected maximum over a sample size of 16.

We measured all three of these adjusted metrics throughout training for a variety of network depths and architectural styles using the batch-averaged gradient direction sampling technique (see Definition 2.7). The results will be discussed in the order the metrics were defined.

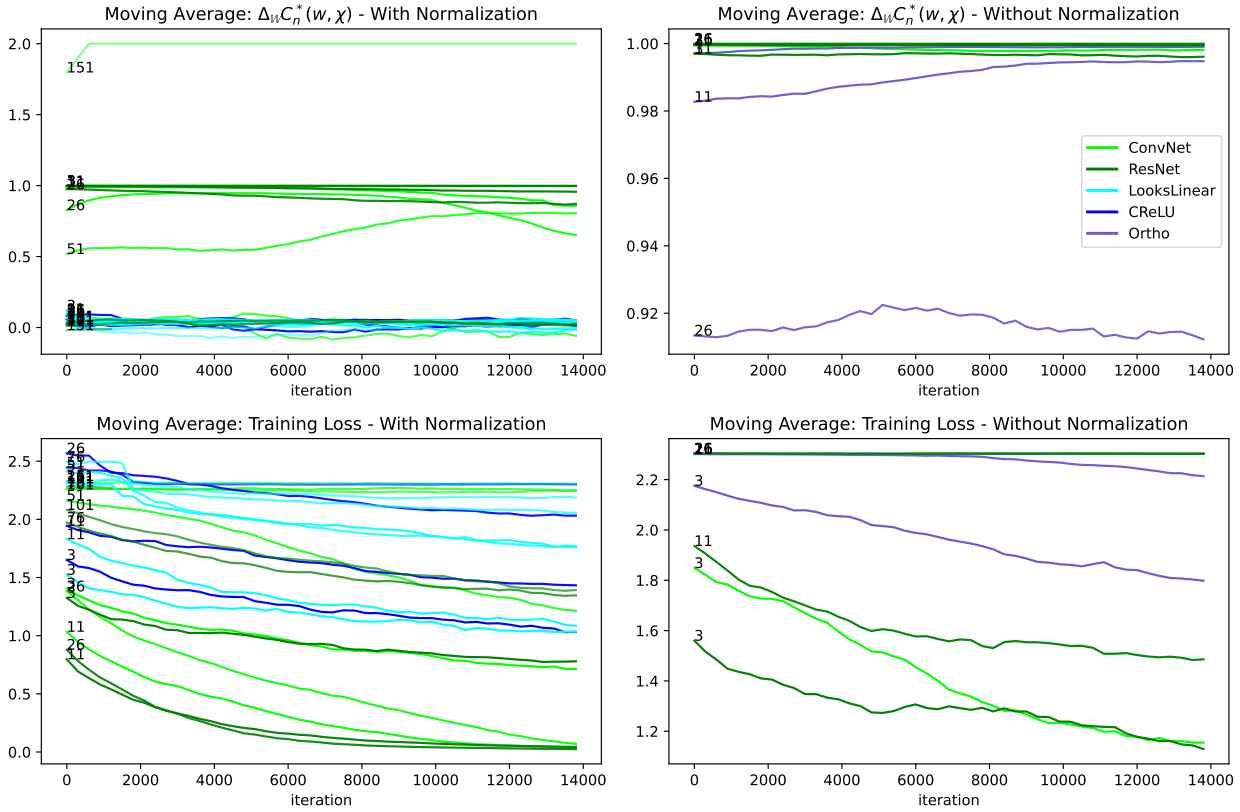


Figure 4.4: The Weight-Gradient Cosine Similarity metric (see Equation 4.1) was measured throughout training for several different types of architectures. There is a strong separation in this metric between architectures tested with and without the Opacus module enabled, so we cannot draw any conclusions without identifying why this strong divergence exists.
The architectures tested included fully connected networks with orthogonal initialization and the ReLU or CReLU activation function, the Looks-Linear architecture (see Definition 3.9), and convolutional networks with and without residual connections. The label on each line is the number of layers in the architecture.

Weight-Gradient Cosine Similarity. Interestingly, there was a stark division between networks that measured high and low in Weight-Gradient Cosine Similarity ($\Delta_{\mathbb{W}} C_n^*(w, \chi)$). After further investigation we determined that the divide was perfectly between networks run in two test batches with slightly modified settings. These settings are completely separate from the calculation of $\Delta_{\mathbb{W}} C_n^*(w, \chi)$, and pertained exclusively to the package Opacus that was used to calculate individual gradient samples efficiently. (Opacus is memory intensive and exhibits a bug in networks using our implementation of the CReLU activation function, which is why these networks were included in a separate test group.) A similar split was not observed in any other metric, and while the ResNets and Convolutional Networks trained in

the second test group (depth greater than 55) have significantly decreased performance, this could be caused by the additional depth and stride adjustments necessary for them to fit on the GPU we used. In any case, this merits further investigation in the future.

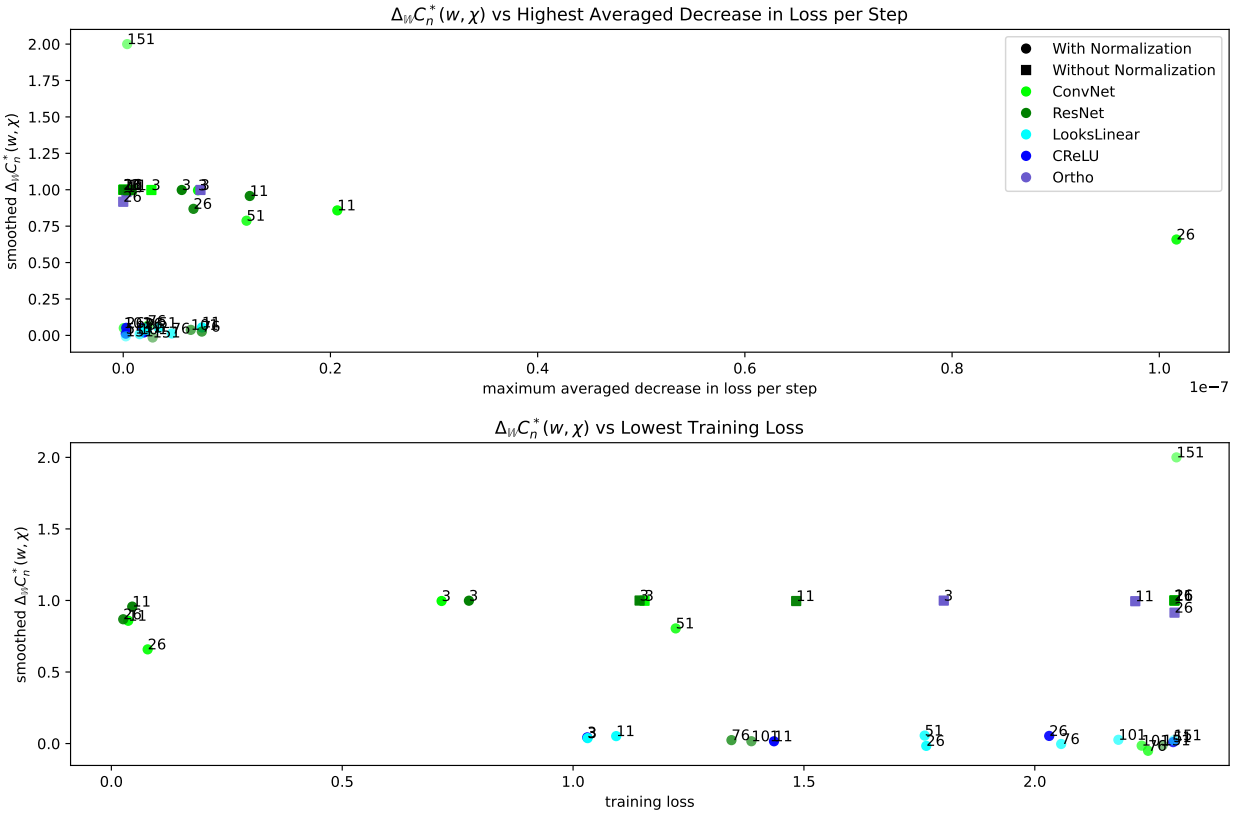


Figure 4.5: The Weight-Gradient Cosine Similarity metric (see Definition 4.1) was measured throughout training for several different types of architectures at the point of steepest descent as well as at the point with the lowest loss (smoothed over a few hundred iterations in each case) and plotted as a point for each network. There is a strong separation in this metric between architectures tested with and without the Opacus module enabled, so we cannot draw any conclusions without identifying why this strong divergence exists.
The architectures tested include fully connected networks with orthogonal initialization and the ReLU or CReLU activation function, the Looks-Linear architecture (see Definition 3.9), and convolutional networks with and without residual connections. The label on each point is the number of layers in the architecture.

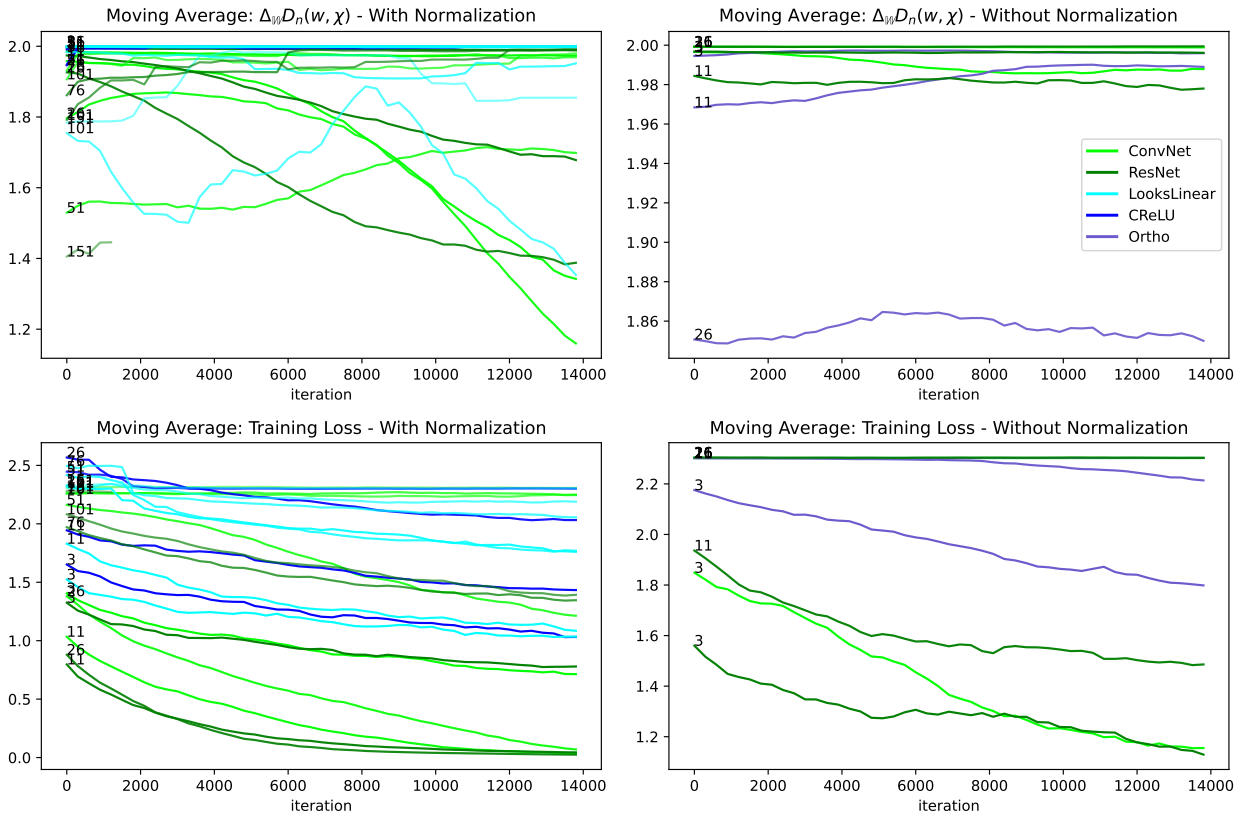


Figure 4.6: The Weight-Gradient Diversity metric (see Equation 4.2) was measured throughout training for several different types of architectures. It is difficult to observe any clear trends in this graph. However, figure 4.7 presents a different view of this data that demonstrates a few interesting correlations.
The architectures tested included fully connected networks with orthogonal initialization and the ReLU or CReLU activation function, the Looks-Linear architecture (see Definition 3.9), and convolutional networks with and without residual connections. The label on each line is the number of layers in the architecture.

Results: Weight-Gradient Diversity Metric. Our measurements of the Weight-Gradient Diversity metric in our training runs show a distinct performance gap between networks that have high and low Weight-Gradient Diversity. While many of the networks with high Weight-Gradient Diversity learned appreciably, every single network that converged near zero loss had quite low Weight-Gradient Diversity measurements near convergence. In other words, only the networks whose Weight-Gradient Diversity measurements decreased during training were able to converge close to zero loss.

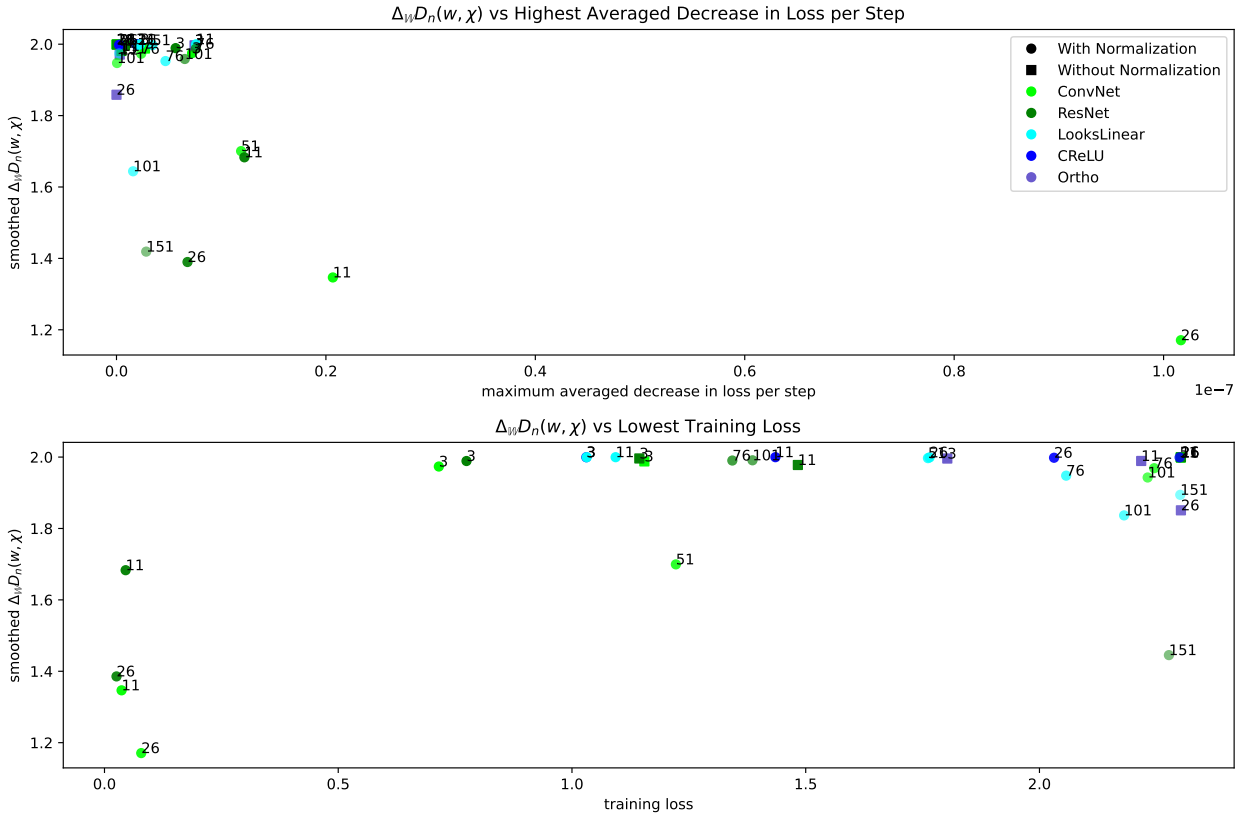


Figure 4.7: The Weight-Gradient Diversity metric (see Definition 4.2) was measured throughout training for several different types of architectures at the point of steepest descent as well as at the point with the lowest loss (smoothed over a few hundred iterations in each case) and plotted as a point for each network. The most apparent feature of these graphs is that there was very little difference in the Weight-Gradient Diversity metric measurements at convergence except for the networks that converged to zero loss (with a few exceptions). It is partially obscured by the scale issue caused by the outlier ConvNet26, but there is also a trend apparent in the point of steepest descent graph.
The architectures tested include fully connected networks with orthogonal initialization and the ReLU or CReLU activation function, the Looks-Linear architecture (see Definition 3.9), and convolutional networks with and without residual connections. The label on each point is the number of layers in the architecture.

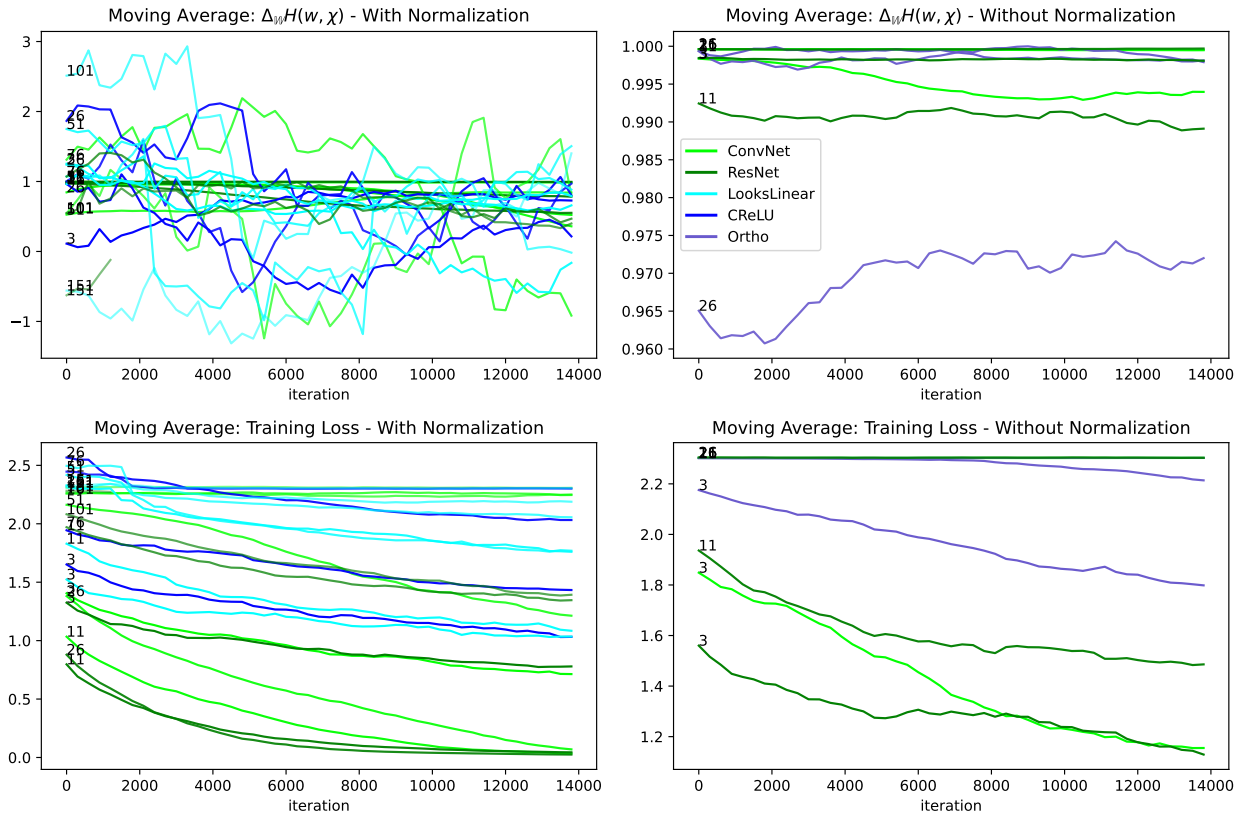


Figure 4.8: The Weight-Gradient Coherence metric (see Equation 4.3) was measured throughout training for several different types of architectures. It is difficult to pick out any clear patterns in this graph due to the number of architectures tested and the amount of noise in the metric. However, figure 4.9 presents the measurements in a way that is more interpretable.
The architectures tested included fully connected networks with orthogonal initialization and the ReLU or CReLU activation function, the Looks-Linear architecture (see Definition 3.9), and convolutional networks with and without residual connections. The label on each line is the number of layers in the architecture.

Results: Weight-Gradient Coherence Metric. Some weak trends are evident in the weight-gradient coherence metric in figure 4.9. Generally speaking, values of weight-gradient coherence between zero and one at convergence correlate generally with a lower final loss value, although that trend is not strong because most values measure close to one with a few outliers that failed to converge appreciably.

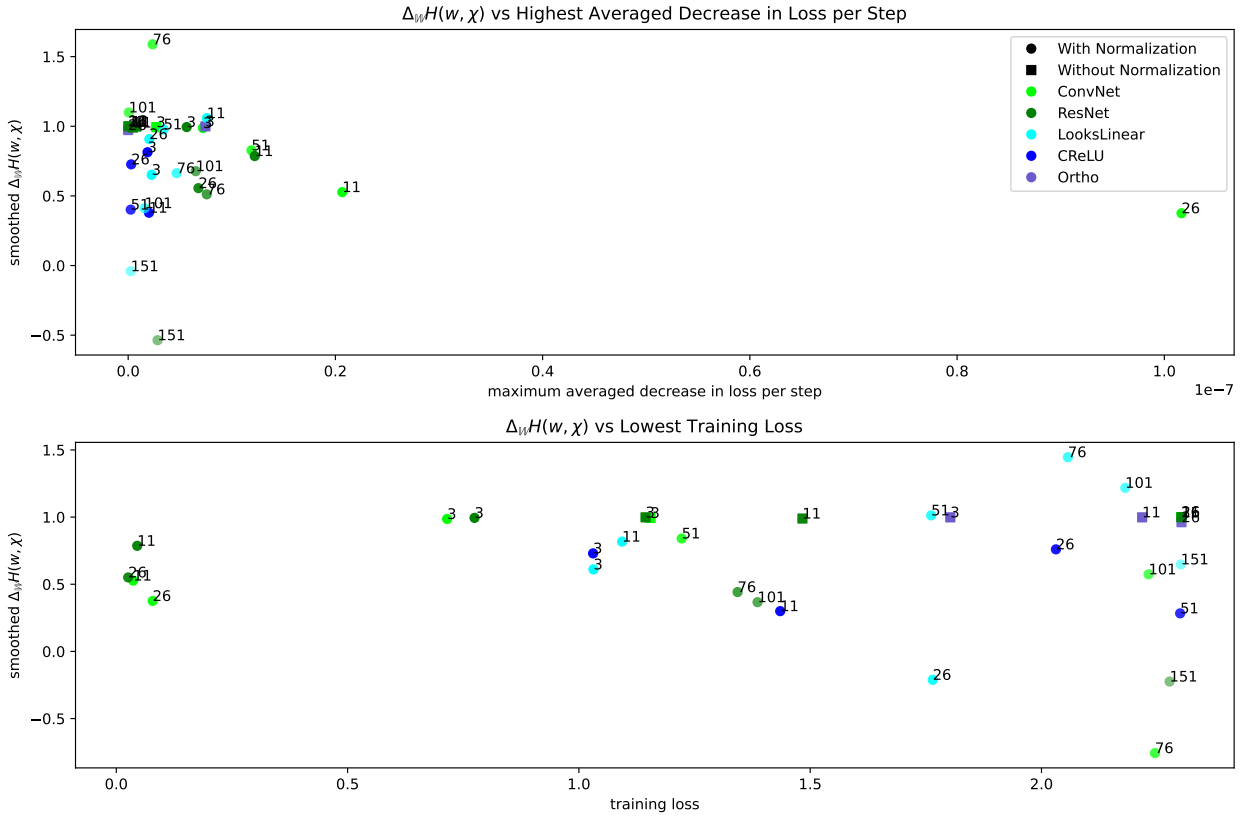


Figure 4.9: The Weight-Gradient Coherence metric (see Definition 4.3) was measured throughout training for several different types of architectures at the point of steepest descent as well as at the point with the lowest loss (smoothed over a few hundred iterations in each case) and plotted as a point for each network. Interestingly, it seems that very low values of Weight-Gradient Coherence are correlated with decreased final performance. Each architecture that converged to near-zero loss had similar measurements at convergence.
The architectures tested include fully connected networks with orthogonal initialization and the ReLU or CReLU activation function, the Looks-Linear architecture (see Definition 3.9), and convolutional networks with and without residual connections. The label on each point is the number of layers in the architecture.

4.2 Future Work: Architecture and Hyper-Parameter Optimization Tests

The fourth and most challenging of our stated objectives was the following:

4. If possible, demonstrate that variance and instability from each source can be reduced or mitigated independently and in a targeted way using architectural or hyperparameter adjustments.

A number of further tests are required in order to accomplish this goal for informed architecture design and optimal hyper-parameter control. We will address these separately in the following two subsections.

4.2.1 Informed Architecture Design Tests. Some of the metrics we measured in our tests showed clear linear trends. In other words, within the scope of our tests learning speed or final training loss was always optimal when that metric was minimized or maximized. However, in several cases the picture was a little more nuanced. In some cases, such as the gradient rank metric with respect to the input $\Delta_{\mathbb{X}} R(h_{\mathbb{X}}, w, \chi)$, higher values were correlated strongly with faster learning at the point of steepest descent in networks that successfully converged (see figure 3.10). However, every network that failed to converge at all also had high measurements in $\Delta_{\mathbb{X}} R(h_{\mathbb{X}}, w, \chi)$. While for most networks $\Delta_{\mathbb{X}} R(h_{\mathbb{X}}, w, \chi)$ decreased as the network was trained, there is a relatively clear boundary in the value of $\Delta_{\mathbb{X}} R(h_{\mathbb{X}}, w, \chi)$ at initialization between networks that failed to converge and those that converged. A few networks managed to converge meaningfully when measuring $\Delta_{\mathbb{X}} R(h_{\mathbb{X}}, w, \chi) > .15$ at initialization, but most failed to do so.

A similar picture occurred with gradient confusion $\Delta_{\mathbb{X}} C_n(w)$ and gradient diversity $\Delta_{\mathbb{X}} D_n(w)$. There was a very strong linear correlation between the best sustained rate of decrease in the loss with decreases in both gradient confusion and gradient diversity in networks that converged at all. However, a number of networks (all without normalization) had very low gradient confusion and low gradient diversity at initialization and failed to converge appreciably.

This is a slightly more nuanced picture than stated in objective 4, however, from our results it appears that optimizing structure in an architecture search or optimizing the learning rate and batch size during training might both be achieved using a mixture of maximization and minimization objectives with boundary constraints over the metrics we have tested.

In order to gain a clearer picture of how different architectural traits impact learning ability and how they perform with regards to the gradient step-consistency and gradient

batch-dissonance metrics we have presented, we will need to expand the number and types of architectures included in our train-to-convergence tests. Namely, other novel and successful architectures should be included, such as DenseNet, Feedback and Direct Feedback Alignment networks, various forms of RNN (including with memory units such as LSTM and RUM), transformer networks, etc.

Tests should also be executed on various modifications to the architectures already tested, such as varying widths, types of normalization, pooling, etc. with an emphasis on modifications where existing theorems and results indicate stronger performance, such as increasing width.

4.2.2 Optimal Hyper-Parameter Control Tests. Because all of our tests thus far have used the same batch-size and learning rate (a decrease on plateau learning rate schedule was used, which means that most of the successfully trained networks all used the same learning rate until near convergence), we cannot say conclusively that any of our metrics could be used to inform an optimal learning-rate or batch-size policy for a given architecture. However, increasing the variety of batch sizes, learning rates, and schedules for both parameters and testing these against the metrics we have presented should reveal whether this is possible.

Optimal Control of the Learning Rate. Early in our research for this thesis, in an attempt to identify a viable target for an optimal learning rate and create a way to validate the correlations already identified in our tests, we tested using the scipy.minimize package to find the distance to the nearest minimum in the loss surface in the direction of the batch-averaged gradient in a ten-layer ResNet at initialization. Unfortunately, we found that this slice of the loss surface was frequently complex enough to make finding the nearest minimum in the direction of the gradient both expensive and inconsistent, even when the same batch was used for every evaluation. However, when a single input was used to evaluate the loss instead of a batch, the slice of the loss surface in the direction of the batch-averaged gradient was not only easy to consistently optimize, it was consistently convex and smooth, appearing

parabolic in a neighborhood centered around the nearest minimum that contained both the starting point and an equal distance in the other direction.

Unfortunately, due to the large amount of variance present in most training sets, the only information we could determine that per-element optimal steps provide about the aggregated loss surface in the direction of the gradient are loose lower and upper bounds on the distance to the closest minimum if it exists.

If, based on our observations about the per-element loss surfaces, we make the following assumptions:

- that the loss surface is parabolic for every input within a batch in the neighborhood containing the nearest minimum,
- and that the closest per-element minimum is always in the direction of the negative batch-averaged gradient rather than the positive batch-averaged gradient,

then the distance to the nearest per-element minimum is smaller than the distance to the nearest minimum of the loss calculated using the batch. This is because, if our assumptions are true, the loss for every element of the batch is uniformly decreasing between the current point and the closest per-element minimum in the direction of the batch-averaged gradient.

However, this was not sufficient information to justify the significant expense of performing a batch of optimizations at every sample point, so we opted against including any metrics based on this in our tests. Furthermore, because this minimization test was performed only at initialization, where convergence is often very rapid for a few iterations, the behavior of the per per-element loss surface in the direction of the batch-averaged gradient could be much less predictable at other points in training.

Moving forward, it seems the most conclusive way to learn what an optimal learning rate policy looks like in relation to the metrics we have presented is to perform a broader range of tests that include variable learning rates and a wider variety of architectures. If these tests revealed connections between the values of any of the metrics that can be efficiently calculated, then the next test required would be to design a learning-rate policy using the

correlated metrics and perform tests using variants of that policy. In order to demonstrate viability of a proposed optimal schedule that is informed by the metrics we have tested, they would need to be tested against final testing performance, wall clock time, etc.

In our own tests we sampled every metric approximately twice per epoch, which increased wall clock time for training runs by a considerable amount for larger models. However, there is considerable room for optimization in our implementations, and it may be possible that near-optimal hyperparameter control would not require multiple samples per epoch. Additionally, a subset of the more efficient metrics we tested may be sufficient for hyperparameter optimization. There was a significant difference in expense across the tested metrics (see figure 4.10 below), with the metrics requiring multiple shifted forward calls and $\Delta_{\mathbb{X}} R(w)$ being significantly more expensive to calculate than the others in large networks.

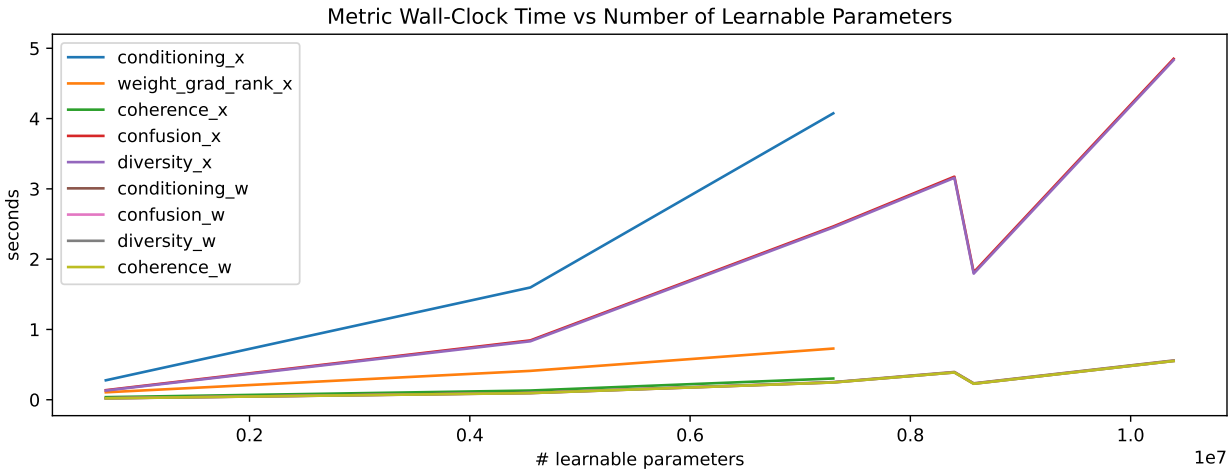


Figure 4.10: Metric Timing by Number of Learnable Parameters

Furthermore, in some cases the value of a metric for a given learning rate or batch size at initialization was highly predictive of convergence. These could be used both to inform architecture design and initial hyperparameter choice without increasing wall-clock time appreciably. Finally, calculating all of the metrics we tested is parallelizable with training if additional GPU's are available, meaning that the training of large models on GPU-clusters could be optimized using a small subset of the cluster.

Optimal Control of the Batch Size. Unfortunately, in practice the batch-size parameter is used much less carefully than the learning-rate parameter. In most cases it is never updated during training. It is true that there is often a large range of viable batch sizes that persists throughout all of training. However, given the evidence we have presented in our literature review and our training tests, there is ample reason to believe that networks could benefit from varying amounts of variance from the batch at different training times. It is well known that small batch-sizes tend to increase final performance with respect to the loss function, while larger batch-sizes result in more stable training. There have been a number of recent papers treating different aspects of this stability-performance trade-off in SGD [16, 17, 24]. In the set of tests included in this thesis only one batch size was used (64), and that decision was motivated primarily by the additional memory requirements imposed by using a larger batch size in parallel with the package Opacus that was used to estimate individual gradients. In order to gain a more complete picture of how the batch-size impacts the various metrics we have presented in this thesis, we will need to perform more tests with a variety of batch sizes.

If enough information is gained from these tests to motivate a policy for updating the batch size during training, it would need to be tested in a variety of networks against final testing performance and speed of training.

4.3 Conclusions

This thesis was motivated primarily by a single observation. The observation that dozens of papers in the field of deep learning could be found approaching the issue of gradient instability from two separate perspectives, many without acknowledging the other. These two perspectives are of course what we have named gradient step-consistency and gradient batch-dissonance. We have cited several papers that analyze gradient step-consistency and smoothness in the loss surface through random projections and analysis of the Hessian. The awareness of gradient batch-dissonance in the field as an issue separate from smoothness in

the loss surface has been growing in the past several years, including the papers we have cited on adversarial attacks, generalization, and particularly the papers that introduced the Relative Effective Rank, Gradient Confusion, Gradient Diversity, and Gradient Coherence metrics. These last papers each present a compelling picture of two distinct ways in which neural networks can exhibit gradient stability.

We have tested these four metrics on a much larger scope than has been performed in any other publication. Furthermore, we introduced six additional metrics that are adaptations of those four or are motivated by our own work with conditioning in chapters 2 and 3. Strong correlations have been demonstrated between several of these metrics and the speed of training and final network performance on the training set. Further tests have been outlined that would not only confirm these observations on a larger variety of architectures and hyperparameter choices, but could also be used in the target for architecture searches (as was done by Guilin et al. using gradient confusion alone [11]) and in designing optimal policies to control the learning-rate and batch-size parameters throughout training.

Moving forward, we hope that this thesis can help unify these two connected, but separate, topics in deep learning, inform more targeted exploration and evaluation of novel architectures, and contribute to more direct and optimal control of hyperparameters during training in deep networks.

Appendix A. Code

All of the code used for this thesis can be found at: `https://bitbucket.org/mvnelson422/gradientconditioning/commits/tag/ThesisComplete`

Bibliography

[1] David Balduzzi et al. "The Shattered Gradients Problem: If resnets are the answer, then what is the question?" In: (2017). DOI: 10.48550/ARXIV.1702.08591. URL: https://arxiv.org/abs/1702.08591.

[2] Satrajit Chatterjee. *Coherent Gradients: An Approach to Understanding Generalization in Gradient Descent-based Optimization*. 2020. DOI: 10.48550/ARXIV.2002.10657. URL: https://arxiv.org/abs/2002.10657.

[3] Satrajit Chatterjee and Piotr Zielinski. *Making Coherence Out of Nothing At All: Measuring the Evolution of Gradient Alignment*. 2020. DOI: 10.48550/ARXIV.2008.01217. URL: https://arxiv.org/abs/2008.01217.

[4] Satrajit Chatterjee and Piotr Zielinski. *On the Generalization Mystery in Deep Learning*. 2022. DOI: 10.48550/ARXIV.2203.10036. URL: https://arxiv.org/abs/2203.10036.

[5] Behrooz Ghorbani, Shankar Krishnan, and Ying Xiao. "An Investigation into Neural Net Optimization via Hessian Eigenvalue Density". In: *Proceedings of the 36th International Conference on Machine Learning*. Ed. by Kamalika Chaudhuri and Ruslan Salakhutdinov. Vol. 97. Proceedings of Machine Learning Research. PMLR, June 2019, pp. 2232–2241. URL: https://proceedings.mlr.press/v97/ghorbani19b.html.

[6] Shuxuan Guo, Jose M. Alvarez, and Mathieu Salzmann. *ExpandNets: Linear Over-parameterization to Train Compact Convolutional Networks*. 2018. DOI: 10.48550/ARXIV.1811.10495. URL: https://arxiv.org/abs/1811.10495.

[7] Gao Huang et al. *Densely Connected Convolutional Networks*. 2016. DOI: 10.48550/ARXIV.1608.06993. URL: https://arxiv.org/abs/1608.06993.

[8] Sergey Ioffe and Christian Szegedy. *Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift*. 2015. DOI: 10.48550/ARXIV.1502.03167. URL: https://arxiv.org/abs/1502.03167.

[9] S. Kaski. "Dimensionality reduction by random mapping: fast similarity computation for clustering". In: *1998 IEEE International Joint Conference on Neural Networks Proceedings. IEEE World Congress on Computational Intelligence (Cat. No.98CH36227)*. Vol. 1. 1998, 413–418 vol.1. DOI: 10.1109/IJCNN.1998.682302.

[10] Sungyoon Lee, Hoki Kim, and Jaewook Lee. *GradDiv: Adversarial Robustness of Randomized Neural Networks via Gradient Diversity Regularization*. 2021. DOI: 10.48550/ARXIV.2107.02425. URL: https://arxiv.org/abs/2107.02425.

[11] Guilin Li et al. *Hierarchical Neural Architecture Search via Operator Clustering*. 2019. DOI: 10.48550/ARXIV.1909.11926. URL: https://arxiv.org/abs/1909.11926.

[12] Hao Li et al. *Visualizing the Loss Landscape of Neural Nets*. 2017. DOI: 10.48550/ARXIV.1712.09913. URL: https://arxiv.org/abs/1712.09913.

[13] Arvind Neelakantan et al. *Adding Gradient Noise Improves Learning for Very Deep Networks*. 2015. DOI: 10.48550/ARXIV.1511.06807. URL: https://arxiv.org/abs/1511.06807.

[14] A. Emin Orhan and Xaq Pitkow. *Skip Connections Eliminate Singularities*. 2017. DOI: 10.48550/ARXIV.1701.09175. URL: https://arxiv.org/abs/1701.09175.

[15] Vardan Papyan. "Traces of Class/Cross-Class Structure Pervade Deep Learning Spectra". In: *CoRR* abs/2008.11865 (2020). arXiv: 2008.11865. URL: https://arxiv.org/abs/2008.11865.

[16] Xin Qian and Diego Klabjan. *The Impact of the Mini-batch Size on the Variance of Gradients in Stochastic Gradient Descent*. 2020. DOI: 10.48550/ARXIV.2004.13146. URL: https://arxiv.org/abs/2004.13146.

[17] Sashank J. Reddi et al. *On Variance Reduction in Stochastic Gradient Descent and its Asynchronous Variants*. 2015. DOI: 10.48550/ARXIV.1506.06840. URL: https://arxiv.org/abs/1506.06840.

[18] Karthik A. Sankararaman et al. *The Impact of Neural Network Overparameterization on Gradient Confusion and Stochastic Gradient Descent*. 2019. DOI: 10.48550/ARXIV.1904.06963. URL: https://arxiv.org/abs/1904.06963.

[19] Shibani Santurkar et al. *How Does Batch Normalization Help Optimization?* 2018. DOI: 10.48550/ARXIV.1805.11604. URL: https://arxiv.org/abs/1805.11604.

[20] Wenling Shang et al. *Understanding and Improving Convolutional Neural Networks via Concatenated Rectified Linear Units*. 2016. DOI: 10.48550/ARXIV.1603.05201. URL: https://arxiv.org/abs/1603.05201.

[21] Mayank Singh, Abhishek Sinha, and Balaji Krishnamurthy. *Neural Networks in Adversarial Setting and Ill-Conditioned Weight Space*. 2018. DOI: 10.48550/ARXIV.1801.00905. URL: https://arxiv.org/abs/1801.00905.

[22] Masato Taki. *Deep Residual Networks and Weight Initialization*. 2017. DOI: 10.48550/ARXIV.1709.02956. URL: https://arxiv.org/abs/1709.02956.

[23] Nicholas J. Teague. *Geometric Regularization from Overparameterization explains Double Descent and other findings*. 2022. DOI: 10.48550/ARXIV.2202.09276. URL: https://arxiv.org/abs/2202.09276.

[24] Chong Wang et al. "Variance Reduction for Stochastic Gradient Optimization". In: *Advances in Neural Information Processing Systems*. Ed. by C.J. Burges et al. Vol. 26. Curran Associates, Inc., 2013. URL: https://proceedings.neurips.cc/paper/2013/file/9766527f2b5d3e95d4a733fcfb77bd7e-Paper.pdf.

[25] Yikai Wu et al. *Dissecting Hessian: Understanding Common Structure of Hessian in Neural Networks*. 2020. DOI: 10.48550/ARXIV.2010.04261. URL: https://arxiv.org/abs/2010.04261.

[26] Yuxin Wu and Kaiming He. "Group Normalization". In: *CoRR* abs/1803.08494 (2018). arXiv: 1803.08494. URL: http://arxiv.org/abs/1803.08494.

[27] Zhuolin Yang et al. *TRS: Transferability Reduced Ensemble via Encouraging Gradient Diversity and Model Smoothness*. 2021. DOI: 10.48550/ARXIV.2104.00671. URL: https://arxiv.org/abs/2104.00671.

[28] Zhewei Yao et al. "PyHessian: Neural Networks Through the Lens of the Hessian". In: (2019). DOI: 10.48550/ARXIV.1912.07145. URL: https://arxiv.org/abs/1912.07145.

[29] Dong Yin et al. *Gradient Diversity: a Key Ingredient for Scalable Distributed Learning*. 2017. DOI: 10.48550/ARXIV.1706.05699. URL: https://arxiv.org/abs/1706.05699.

[30] Alireza Zaeemzadeh, Nazanin Rahnavard, and Mubarak Shah. *Norm-Preservation: Why Residual Networks Can Become Extremely Deep?* 2018. DOI: 10.48550/ARXIV.1805.07477. URL: https://arxiv.org/abs/1805.07477.