Signal propagation and Initialization in Deep Neural Networks

A Thesis

submitted to
Indian Institute of Science Education and Research Pune
in partial fulfillment of the requirements for the
BS-MS Dual Degree Programme

by

Dayal Singh

IISER PUNE

Indian Institute of Science Education and Research Pune
Dr. Homi Bhabha Road,
Pashan, Pune 411008, INDIA.

June, 2021

Supervisor: Dr. Sreejith G J

Certificate

This is to certify that this dissertation entitled Signal propagation and Initialization in Deep Neural Networks towards the partial fulfilment of the BS-MS dual degree programme at the Indian Institute of Science Education and Research, Pune represents study/work carried out by Dayal Singh at Indian Institute of Science Education and Research under the supervision of Dr. Sreejith G J, Assistant Professor, Department of Physics , during the academic year 2020-2021.

Dr. Sreejith G J

Committee:

Dr. Sreejith G J

Prof. Deepak Dhar

This thesis is dedicated to all the people we lost in the COVID-19 pandemic.

Declaration

I hereby declare that the matter embodied in the report entitled Signal propagation and Initialization in Deep Neural Networks are the results of the work carried out by me at the Department of Physics, Indian Institute of Science Education and Research, Pune, under the supervision of Dr. Sreejith G J and the same has not been submitted elsewhere for any other degree.

[illegible]

Dayal Singh

Acknowledgments

I would like to express my deep and sincere gratitude to my supervisor, Dr. Sreejith G J, for giving me an opportunity to work on my research interest and his guidance throughout this thesis project. He has trained me in research methodology and helped in presenting research works as clearly as possible.

I am extremely grateful to my mother and family for their continuous support throughout the thesis work. I would also like to express my gratitude to my friends Rakshitha, Omkar, and Manraj for their constant encouragement and support.

I want to express my gratitude to my thesis expert, Prof. Deepak Dhar, for his helpful inputs in the project; Prof. MS Madhusudhan for his unconditional support throughout my IISER journey; and Prof. MS Santhanam for his constant encouragement. Apart from them, I would also like to thank Prof. Bhas Bapat, Prof. Sunil Mukhi, Prof. Bijay Kumar Agarwalla, Prof. Prasad Subramanian, Prof. Sutirth Dey, and all other IISER Pune faculty for supporting me throughout my IISER journey.

Last but not least, I also want to express my gratitude to the computing facilities at IISER Pune that made this research possible. I have spent most of my IISER life on IISER computing resources, including Param Brahma, and I would like to express my deepest gratitude to them for not giving up on me. I would also like to thank Nisha Kurkure and Neeta Deo from the IT department for helping me out with the cluster facilities at IISER Pune.

Abstract

Despite their successful application in various fields of science and technology, deep neural networks remain poorly understood. The ability of overparameterized neural networks to express complex functions (expressivity) is one of the major theoretical questions, among others. Furthermore, the expressivity of a deep neural network at the initialization is a crucial theoretical aspect due to the use of local (gradient-based) algorithms for training, which can be analyzed by considering signal propagation in infinitely wide networks (mean-field limit) as a model. This mean-field analysis suggests that deep neural networks have an ordered and a chaotic phase, and they achieve exponential expressivity as a function of depth in the chaotic phase. However, signals become highly correlated in deep ReLU (Rectified Linear Unit) networks with uncorrelated weights due to the non-existence of a chaotic phase; this suggests that deep ReLU networks have low expressive power. Using the mean-field theory of signal propagation, we analyze the evolution of correlations between signals propagating through a ReLU network with correlated weights. Furthermore, we show that ReLU networks with anti-correlated weights can avoid this low expressivity outcome and have a chaotic phase where the correlations saturate below unity. Consistent with this analysis, we find that networks initialized with anti-correlated weights can train faster by taking advantage of the increased expressivity in the chaotic phase. Combining this with a previously proposed strategy of using an asymmetric initialization to reduce dead node probability (probability of the propagated signals reaching a low sensitivity domain of the ReLU activation function), we propose an initialization scheme that allows faster training and learning than other initialization schemes on various tasks.

Contents

Tables

List of plots

Chapter 1

Introduction

Machine learning is a set of algorithms designed to discover meaningful structure in data by experience with applications to many domains of science and technology. They have shown remarkable results in computer vision[1, 2, 3], speech recognition[4, 5, 6], intelligent gaming[7], analyzing particle data [8], drug discovery [9], and protein folding [10], to name a few. The credit of the remarkable success of machine learning in the last decade mostly goes to deep learning, which concerns learning useful features (represenations of data) from the raw input data on its own. In contrast, conventional machine learning methods require a careful construction of useful representations to perform complicated tasks; it limits their applications as they require engineering a feature extractor that needs expertise in various domains. On the other hand, deep learning machines learn meaningful representations independently and do not require engineering an external feature extractor.

A deep learning machine consists of multiple layers of representations of the raw data in increasing levels of abstraction. The simplest example of a deep learning machine is a deep neural network shown in Figure 1.1. At each layer, a non-linear operation with learnable parameters is applied to achieve high expressivity. The parameters are trained using a chain rule of derivatives, called the backpropagation algorithm. The high-level representations learned are capable of differentiating various patterns and suppressing noise within the raw data. Other architectures include deep convolutional neural networks (CNNs), [11] and recurrent neural networks (RNNs) [12], which are specialized neural networks for image and text data. They have shown remarkable success in image recognition, audio and video processing, speech recognition, and language trans-

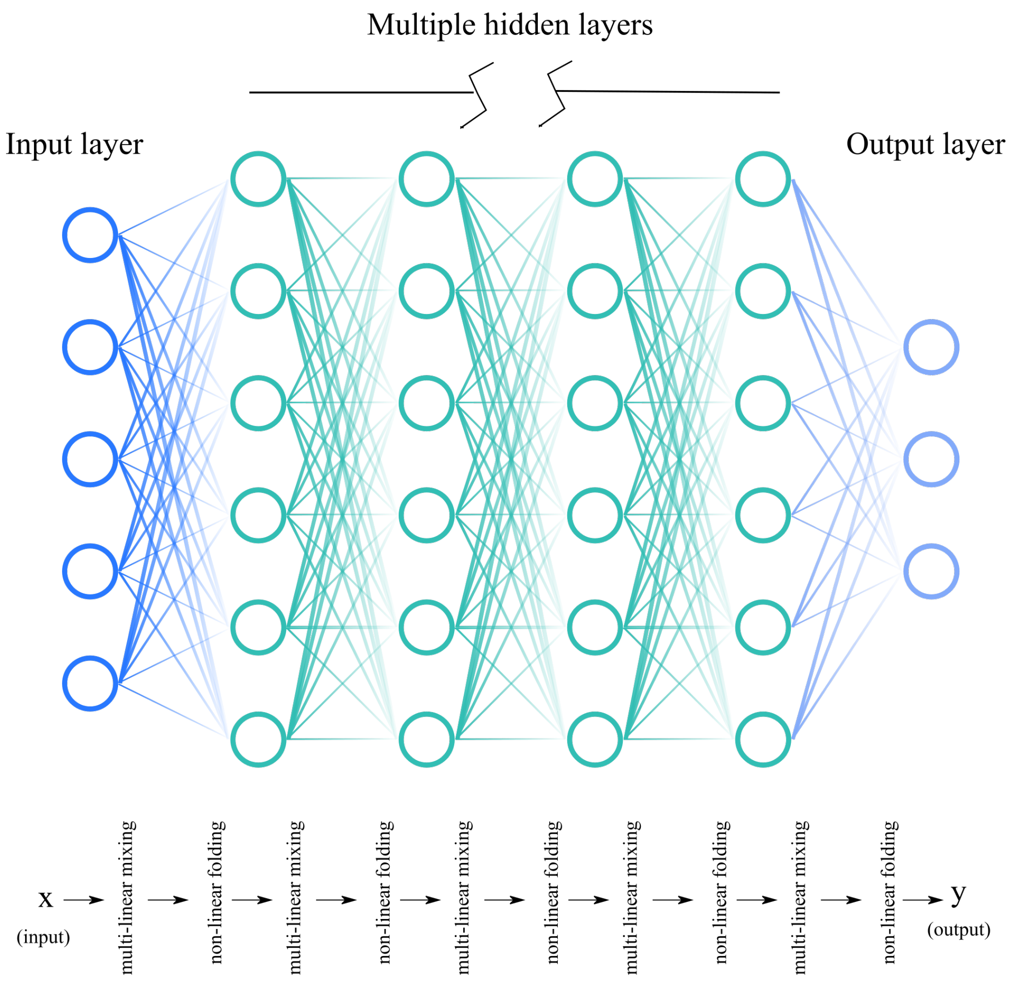


Figure 1.1: A deep neural network with multiple layers of neurons stacked together. It has three major components —an input layer to which input data is fed, an output layer that outputs the generated signal, and multiple hidden layers that transform the input into high-level representations.

lations. In this work, we will focus on deep neural networks; for further reading about CNNs and RNNs, we suggest Ref. [13].

The most common form of machine learning is supervised learning, which involves learning input-output maps/functions from examples. Supervised learning improves upon Reinforcement learning by guiding the machine by directional improvement for the parameters rather than just a boolean response. The first step in supervised learning is to randomly initialize the machine with learnable parameters, often called weights. Next, consider a fixed set of input-output pairs, called a training set. As the machine is untrained, it computes random output for a given input dataset. An average error (over the training set), often called loss or cost, of the desired and machine-generated output is computed, and the learnable parameters are tuned to minimize this loss function. It is a highly non-convex optimization problem with millions of parameters and no assurance of attaining the global minimum. We can think of this process as navigating the system to a reasonable minimum of the high-dimensional loss landscape (see Figure 1.2 for a pictorial representation in one dimension). The parameters are updated using a gradient-based algorithm like Stochastic Gradient Descent (SGD), which adjusts the weights by moving towards the opposite direction to the gradient vector at the current coordinates. We repeat the process from this newly arrived coordinates (weights here) till we reach a reasonable minimum of the loss landscape. Finally, we measure the generalization performance, a measure of performance on unseen data, of the learned map on new examples by calculating the loss for a different set of examples, often called a testing set.

The training procedure involves making several choices, which can completely change the training dynamics. It includes choosing the number of layers, non-linear operation at each layer, distribution for parameter initialization, loss function, optimization algorithm, and training time. These are called hyperparameters, which can be selected by optimizing them over a set of examples, called a validation set. We can calculate the generalization using the testing set, as usual, for the machine with optimal hyperparameters. Often, it is not feasible to obtain the optimal hyperparameters by sheer brute force. Theoreticians and practitioners provide some guidelines to select these hyperparameters, but there remain several open-end questions to be answered.

Despite their success in various fields, deep learning algorithms are not well understood and are often treated as black boxes. Even the simplest of deep learning algorithms, feed-forward neural networks, lack a mathematical foundation. The ability of overparameterized neural networks to express complex functions (expressivity) is one of the central theoretical questions, among others.

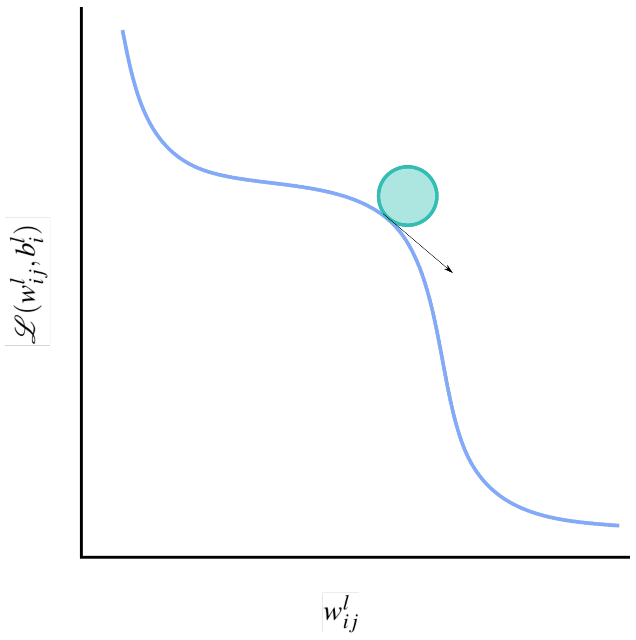


Figure 1.2: Pictorial representation of the optimization of the loss function $\mathscr{L}(w^l_{ij}, b^l_i)$ (y axis) with respect to one of its parameters denoted by w^l_{ij} (x axis). The loss function measures the distance between the desired and network generated output. The neural network parameters (coordinates of this loss landcape) are updated by moving in a direction opposite to the gradient vector at the current coordinates. This process is repeated until convergence. In general, a neural network landscape has an order of $\mathscr{O}(N^2L)$ parameters, where N is the width and L is the depth of the network.

An ensemble of networks with lower expressivity can be thought of as having high correlations between two different output signals, whereas networks with higher expressivity will have lesser correlations between any two output signals. Furthermore, the expressivity of a deep neural network at the initialization is a crucial theoretical aspect due to the use of local (gradient-based) algorithms for training, which can be analyzed by considering signal propagation in infinitely wide networks (mean-field limit) as a model. This mean-field analysis suggests that deep neural networks have an ordered and a chaotic phase, and they achieve exponential expressivity as a function of depth in the chaotic phase. However, signals become highly correlated in deep neural networks with ReLU activation (see Figure 1.3) initialized with uncorrelated Gaussian weights. Due to the non-existence of a chaotic phase, deep ReLU networks have low expressive power. However, in practice, they approximate complex functions. Therefore, in this thesis, we focus on the mean-field theory of signal propagation through deep ReLU networks and its implications on initializing them for better performance.

We organize the thesis as follows. First, we begin by reviewing deep neural networks in Section 1.1 and describe the signal propagation problem and its application to initializing deep neural networks. Next, we review the features and complexities of deep ReLU networks in Section 1.2, which is the central focus of this work.

The second chapter describes the methods used to analyze deep neural networks and the training tasks to validate the new results of this work. Section 2.1 reviews a simple method to derive the mean-field equations for signal propagation. Section 2.2 reviews an alternate method based on a path integral approach to derive the mean-field equations and their applications for studying perturbations applied to a deep neural network. Lastly, Section 2.3 describes various tasks based on the teacher-student setup used to validate the new results of this work.

The third chapter introduces a new initialization scheme for deep ReLU networks and demonstrates its advantage over other initialization schemes. Section 3.1 analyzes mean-field equations for signal propagation with correlated weights and introduces Anti-Correlated Initialization (ACI) for ReLU networks. Section 3.2 studies the training dynamics and performance of ReLU networks with correlated weights. Section 3.3 combines ACI with a previously proposed strategy of using asymmetric weights and introduces Random Asymmetric Anti-correlated Initialization (RAAI), which is the main result of this thesis. Section 3.4 compares various initialization schemes for ReLU networks over training tasks described in Section 2.3. Lastly, chapter 4 concludes this thesis by discussing various aspects of this new initialization scheme.

1.1 Deep Neural Networks

A feed-forward neural network with multiple layers is the simplest example of a deep learning machine. It consists of neuron layers stacked together, as shown in Figure 1.1. The value at each node, called an activation, is obtained by considering a multi-linear transformation of the last layer, followed by a non-linear activation function. The value at the node before applying the non-linear activation is called a pre-activation. The input signal propagates left to right, going through an alternate sequence of multi-linear expansion and non-linear folding, each having learnable parameters. The output of a neural network can be continuous or categorical.

Despite its success, the construction of a successful deep neural network remains highly practiced, with no clear conceptual basis [14]. It raises various theoretical questions about how and when they work. For example, is deeper network always better? How information propagates through deep neural networks? What are the necessary conditions to train them? Why and when do they generalize? How SGD attains a good solution avoiding local minima? In this thesis, we aim to understand a few of these questions from the perspective of statistical physics and information theory. In particular, we explore the signal propagation problem in deep neural networks and its application weight initialization.

Consider a feed-forward neural network with L layers with N_l neurons in layer l. For an input signal x from a training set $\mathscr{T}$, we denote the pre-activation at node i in layer l by $h_i^l(x)$ and activation by $s_i^l(x)$. Here, we write $s_i^l(x)$, $h_i^l(x)$ to show the explicit dependence of the activation, pre-activation on the input signal x. A signal $(s_1^{l-1}, \dots, s_{N_l}^{l-1})$ at layer $l-1$ propagates to layer l by the rule

$$\begin{aligned} h_i^l(x) &= \sum_{j=1}^{N_{l-1}} w_{ij}^l s_j^{l-1}(x) + b_i^l \\ s_i^l(x) &= \phi(h_i^l(x)), \end{aligned}$$

where ϕ is a non-linear activation function and w_{ij}^l, b_i^l are tunable parameters called weights and biases. ReLU, $\phi(x) = \max(0, x)$, is the most widely used activation function; other choices include sigmoid, tanh, and swish. Figure 1.3 shows some common activation functions. A common feature of all these continuous functions is that they are many-to-one and tend to fold the input space. Moreover, they are inspired mainly by physical systems. For example, asymmetric ones (ReLU and swish) mimic non-linear threshold dynamics, whereas symmetric functions (sigmoid and tanh) are inspired by complex systems that start slowly, accelerate and saturate over time (like bacterial growth curve).

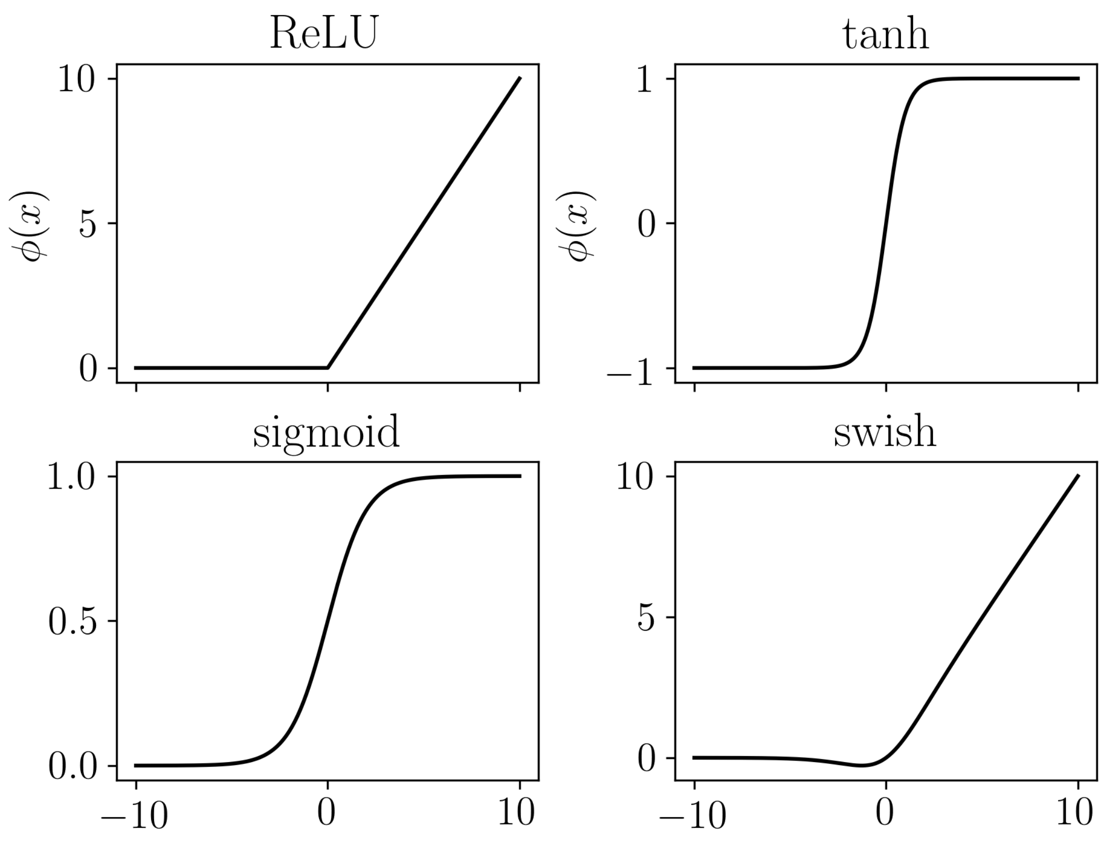


Figure 1.3: Some commonly used activation functions for deep neural networks.

In this work, we will restrict ourselves to Gaussian distribution for initializations. Throughout this thesis, we will denote Gaussian distriibution with mean μ and variance σ^2 as $\mathscr{N}(\mu, \sigma^2)$. Using this notation, the weights and bias distributions are $w^l_{ij} \sim \mathscr{N}(0, \frac{\sigma_w^2}{N_{l-1}})$, $b^l_i \sim \mathscr{N}(0, \sigma_b^2)$. This reduces the parameter distribution to have only two tunable hyperparameters, σ_w^2, and σ_b^2. The later sections show that only a subspace of these parameters ensures optimal information flow through the network. Note that the $1/N_{l-1}$ scaling in the weight's variance ensures that the input contribution from the last layer to each neuron is $\mathscr{O}(1)$ for large width N_{l-1}.

Next, we consider a loss function $\mathscr{L}(w^l_{ij}, b^l_i)$, which measures the distance between the generated and desired output. For continuous outputs, mean-squared error (MSE), and for categorial output, sparse cross-entropy are common choices. Initially, the weights and bias are random, and consequently, the output is random as well. Therefore, the average loss is initially large, and we aim to tune the weights and bias to minimize this loss. An ideal initialization would start with a tiny loss and have a straightforward path towards the minima. Furthermore, we calculate gradient vectors for the weights $\frac{\partial \mathscr{L}(w_{ij}^l, b_i^l)}{\partial w_{ij}^l}$ and bias $\frac{\partial \mathscr{L}(w_{ij}^l, b_i^l)}{\partial b_i^l}$ using the loss function at the current coordinates $\{w_{ij}^l, b_i^l\}$ of the loss landscape, using chain-rule of derivatives (known as backpropagation algorithm), and move the weights in a direction opposite to this gradient vector. This procedure is summarised pictorially in Figure 1.2. Stochastic gradient descent is a commonly used update rule for updating the parameters given by

$$\begin{aligned} w_{ij}^l &= w_{ij}^l - \eta \frac{\partial \mathscr{L}(w_{ij}^l, b_i^l)}{\partial w_{ij}^l} \\ b_i^l &= b_i^l - \eta \frac{\partial \mathscr{L}(w_{ij}^l, b_i^l)}{\partial b_i^l}, \end{aligned} \tag{1.1}$$

where η is the step size (called learning rate), and the partial derivates are the averaged gradient vectors over the training set. The averaged gradient vector is usually calculated for a randomly picked subset of the training set, called a mini-batch. Estimating the gradient vector for a mini-batch reduces the matrix size, which speeds up computations. Also, the randomness introduced due to the mini-batch procedure may help in escaping local minima. Other popular optimizer choices include Adam [15], RMSprop [16], and Adadelta [17], which improve on this procedure. Adadelta and RMSprop normalize the gradients in Eqn. 1.1 by their exponentially weighted root mean squared value to make a similar size update in all directions. This process can be viewed as making the loss landscape symmetrical around the local minima. Adam further improves on Adadelta and RMSprop by considering an exponentially weighted average of the gradients in Eqn. 1.1, which can be viewed as imparting momentum to the system. For a review on gradient-based optimizers, we suggest reading Ref. [18]. Next, we update the weights till convergence; the number of times the optimization procedure goes through the entire dataset is called an epoch. Finally, we calculate the generalization performance on an unseen testing dataset.

It may be of concern that these local algorithms will get stuck in local minima, but they are pretty efficient at finding reasonably good solutions. There are two lines of thought for this. First, in high dimensions, the probability of encountering a local minimum is relatively low unless we are already at the bottom [14]. Thus, most of the fixed points encountered are saddle points. Second, the randomness induced by using a mini-batch procedure helps escape local minima when encountered. Moreover, the solutions found by SGD generalize well to unseen examples. Multiple studies hint that due to its stochastic nature, SGD is biased towards flat minima, which are less prone to changes in the loss landscape due to the finite size of the training set [19]. Nevertheless, it is challenging to train deep networks with millions of parameters with gradient-based algorithms [20, 21]. These optimizers suffer from vanishing and exploding gradient problems. Sometimes, the backpropagating gradient becomes too tiny and prevents the weights from changing, leading to a perpetually inactive state. This is called the vanishing gradient problem. On the other hand, the exploding gradient problem occurs when the gradients become too large, leading to drastic changes in the weight update, leading to failure in learning. These problems are more profound in deeper networks as repeated non-linear operations might lead to vanishing or explosion of the input signal.

Several methods are proposed to overcome the vanishing/exploding gradient problem; these can be classified into three categories [22]. The first approach modifies the architecture, which includes using modified activation functions [23, 3, 24, 25, 26, 27], adding connections between non-consecutive layers (residual connections)[28], and optimizing network depth and width. However, the proposed activations are often computationally less efficient and require a fine-tuned parameter [22]. The second approach relies on normalization techniques [29, 30, 31, 32, 33], in which either the input signals or weights are normalized after some steps to ensure that the signal and the corresponding gradients do not vanish or become large. For example, we can normalize the input signals to have zero mean and unit variance after propagating through each layer. The most popular normalizing technique is batch normalization [30]. Batch normalization prevents the vanishing and exploding gradients by normalizing the output at each layer but with an additional computational cost of up to 30% [34]. A related strategy involves using the self-normalizing activation (SeLU), which ensures output with zero mean and unit variance by construction [26]. The third approach focuses on the initialization of the weights and biases. A careful initialization may avoid vanishing and exploding gradient problems and help the system to find a reasonable minimum relatively quickly [35]. He-initialization [3] for ReLU activation and Glorot-initialization [36]for sigmoid type activations are commonly used strategies. In Glorot initialization, weights are drawn from $\mathcal{N}(0, 1/N)$, where N is the width of the network. The $1/N$ scaling factor ensures that the length of the signal does not become too large or too small after repetitive application of the weight matrix in a deep network. He initialization specializes this approach to ReLU activation. The extra factor of two takes into account that ReLU outputs a non-zero value only for positive inputs.

A growing body of work has analyzed infinitely wide networks (mean-field limit) as a model to study deep neural networks [37, 38, 39, 40, 41, 42, 43, 44]. In particular, Ref. [38, 40] studies the signal propagation problem and suggests conditions for optimal flow through a network.

Furthermore, initializing the network at these optimal parameters avoids vanishing and exploding gradient problems while training. This thesis aims to improve the initialization scheme for deep neural networks with a ReLU activation guided by the mean-field theory of signal propagation. In the following section, we review the features and complexities of deep ReLU networks.

1.2 Deep ReLU Networks

ReLU, $\phi(x) = max(0,x)$, outperforms most of the other activation functions proposed [36]. It has several advantages over other activations. First, the ReLU activation function is computationally simple as it essentially involves only a comparison operation. Second, ReLU suffers less from vanishing gradients, a major problem in training networks with sigmoid-type activations that saturate at both ends [1]. For a large value of the input ($|x| > 4.5$), sigmoid and tanh activation saturate (see Figure 1.3), and their first derivative is zero, encountered while updating weights (see Eqn.1.1). Finally, ReLU networks are known to generalize well even in the overly parameterized regime [45], which does not fit our intuition from polynomial regression.

Despite its success, ReLU also has a few drawbacks, one of which is the dying ReLU problem [3, 24]. The dying ReLU is a type of vanishing gradient problem in which the network outputs zero for all inputs (all pre-activations are non-positive) and is dead. There is no gradient flow in this state. Recently, Ref. [22] proposed Random Asymmetric initialization (RAI), which reduces the probability of dead node at the initialization. ReLU also suffers from exploding gradient problem sometimes, which occurs when backpropagating gradients become large [46]. In this work, we aim to improve the initialization scheme for ReLU networks using the mean-field theory of signal propagation.

Multiple studies analyzed ReLU networks in the large width limit, attempting to unravel mysteries about ReLU activations. For example, Ref.[42] showed that correlations in input signals propagating through a ReLU network always converge to one. Many other works found that ReLU networks are biased towards simpler functions [47, 48, 49, 50, 51], which may account for their better generalization properties even in the overly parameterized regime. However, from their successful application in different domains, one may guess that they should be capable of computing more complex functions. Thus, there might be a subspace of the parameters where the network can represent complex functions.

Ref. [43, 44] applied weight and input perturbations to analyze the function space of ReLU networks. They found that ReLU networks with anti-correlated weights explore the function space farther than uncorrelated/positively correlated weights. Ref. [52] found that ReLU CNN's produce anti-correlated feature matrices after training. These two studies motivated us to analyze the correlation properties of signal propagation in anti-correlated ReLU networks.

Following the mean-field theory of signal propagation proposed by Ref. [38], we found that ReLU networks with anti-correlated weights have a chaotic phase, which implies higher expressivity. In contrast, ReLU networks with uncorrelated weights do not have a chaotic phase. Furthermore, we find that initializing ReLU networks with anti-correlated weights results in faster training. We call it Anti-correlated initialization (ACI). Further performance improvement is achieved by incorporating RAI, which reduces the dead node probability. This combined scheme, which we call Random anti-correlated asymmetric initialization (RAAI), is the main result of this work and is defined as follows. We pick weights and bias incoming to a neuron from anti-correlated Gaussian distribution (like Eqn. 3.1) and replace a randomly picked weight/bias with a random variable drawn from a beta distribution. The code to generate weights drawn from the RAAI distribution is given in Appendix D. We analyze the correlation properties of RAAI and show that it performs better than the best-known initialization schemes on tasks of varying complexity. It may be of concern that initialization in an expressive space may lead to overfitting, and we do observe the same for ACI for some tasks. In contrast, RAAI shows no signs of overfitting and performs consistently better than all other initialization schemes.

The next chapter describes the methods used to quantify the above results. We begin by reviewing the mean-field theory of signal propagation, and later, we describe the training tasks used to quantify the performance of different initialization schemes.

Chapter 2

Methods

This chapter describes the theoretical and numerical methods used in this work. The first two sections review the mean-field theory of signal propagation in deep neural networks using two methods. First, we derive the mean-field equations, which describe how the correlations between input signals evolve while propagating through a deep neural network. Next, we describe another method, based on generating functionals, to derive the same equations. The latter approach is more general and can be used to study perturbations applied to deep neural networks. The last section describes the various training task used to compare proposed initialization schemes with previously proposed ones.

2.1 Mean-field theory of signal propagation in deep neural networks

We begin by reviewing a mean-field approach proposed by Ref. [38, 40] to analyze signal propagation through infinitely wide neural networks. In the next chapter, we modify this mean-field analysis to analyze ReLU networks with anti-correlated and asymmetric weights.

Consider a fully connected neural network with L layers with N_l neurons in layer l. For an input signal x, we denote the pre-activation at node i in layer l by $h_i^l(x)$ and activation by $s_i^l(x)$. A signal $(s_1^{l-1}, \ldots, s_{N_l}^{l-1})$ at layer $l-1$ propagates to layer l by the rule

$$\begin{aligned} h_i^l(x) &= \sum_{j=1}^{N_{l-1}} w_{ij}^l s_j^{l-1}(x) + b_i^l \\ s_i^l(x) &= \phi(h_i^l(x)), \end{aligned} \tag{2.1}$$

where ϕ is the non-linear activation function and $w_{ij}^l \sim \mathcal{N}(0, \frac{\sigma_w^2}{N_{l-1}})$, $b_i^l \sim \mathcal{N}(0, \sigma_b^2)$ are the weights and biases. Note that the $1/N_{l-1}$ scaling in the weight's variance ensures that the input contribution from the last layer to each neuron is $\mathcal{O}(1)$.

To track the layer-wise information flow, consider the normalized length and overlap of the pre-activations for two input signals x_1, x_2, reaching layer l

$$\begin{aligned} q_h^l(x_a) &= \frac{1}{N_l} \sum_{i=1}^{N_l} \left(h_i^l(x_a)\right)^2 \text{ where } a \in \{1,2\} \\ q_h^l(x_1,x_2) &= \frac{1}{N_l} \sum_{i=1}^{N_l} h_i^l(x_1) h_i^l(x_2). \end{aligned}$$

The covariance matrix between the signals is obtained by performing appropriate averages of q_h^l over input signals.

Assuming self averaging, consider an average over all possible networks, i.e., the weights and biases. For simplicity of notations later, we use the same symbol for averaged q_h^l.

$$\begin{aligned} q_h^l(x_a) &= \frac{1}{N_l} \left\langle \sum_{i=1}^{N_l} (h_i^l(x_a))^2 \right\rangle = \sum_{j,k=1}^{N_{l-1}} \left\langle w_{ij}^l w_{ik}^l \right\rangle \phi(h_j^{l-1}(x_a) \phi(h_k^{l-1}(x_a) + \left\langle (b_i^l)^2 \right\rangle, \\ q_h^l(x_1,x_2) &= \frac{1}{N_l} \left\langle \sum_{i=1}^{N_l} h_i^l(x_1) h_i^l(x_2) \right\rangle = \sum_{j,k=1}^{N_{l-1}} \left\langle w_{ij}^l w_{ik}^l \right\rangle \phi(h_j^{l-1}(x_1)) \phi(h_k^{l-1}(x_2)) + \left\langle (b_i^l)^2 \right\rangle, \end{aligned}$$

where $\langle .. \rangle$ denotes an average over the weights and biases. Since the weights and biases are uncorrelated, we obtain

$$
\begin{aligned}
q_h^l(x_a) &= \frac{\sigma_w^2}{N_{l-1}} \sum_j^{N_{l-1}} \left(\phi(h_j^{l-1}(x_a)\right)^2 + \sigma_b^2 \\
q_h^l(x_1,x_2) &= \frac{\sigma_w^2}{N_{l-1}} \sum_j^{N_{l-1}} \phi(h_j^{l-1}(x_1))\phi(h_k^{l-1}(x_2)) + \sigma_b^2.
\end{aligned}
$$

For large N_{l-2}, each $h_i^{l-1}(x)$ is a weighted sum of uncorrelated random variables (see Eqn. 2.1). Thus, we expect the distribution of $h_i^{l-1}(x_1), h_i^{l-1}(x_2)$ to converge to a zero-mean Gaussian with a covariance matrix given by

$$
\Sigma_h^{l-1}(x_1,x_2) = \begin{bmatrix} q_h^{l-1}(x_1) & q_h^{l-1}(x_1,x_2) \\ q_h^{l-1}(x_1,x_2) & q_h^{l-1}(x_2) \end{bmatrix}.
$$

On replacing the average over nodes by Gaussian distribution with appropriate covariance, we obtain iterative maps for the length and overlap

$$
\begin{aligned}
q_h^l(x) &= \mathscr{V}\left(q_h^{l-1}(x) \mid \sigma_w^2, \sigma_b^2\right) \\
q_h^l(x_1,x_2) &= \mathscr{C}\left(c_h^{l-1}(x_1,x_2), q_h^{l-1}(x_1), q_h^{l-1}(x_2) \mid \sigma_w^2, \sigma_b^2\right),
\end{aligned}
\tag{2.2}
$$

where the map $\mathscr{V}$ is given by

$$
\mathscr{V}\left(q_h^{l-1}(x) \mid \sigma_w^2, \sigma_b^2\right) = \sigma_w^2 \int Dz\, \phi\left(\sqrt{q_h^{l-1}(x)}\, z\right)^2 + \sigma_b^2, \tag{2.3}
$$

and the map $\mathscr{C}$ is given by

$$
\begin{aligned}
&\mathscr{C}\left(c_h^{l-1}(x_1,x_2), q_h^{l-1}(x_1), q_h^{l-1}(x_2) \mid \sigma_w^2, \sigma_b^2\right) = \sigma_w^2 \int Dz_1 Dz_2\, \phi(u_1)\phi(u_2) + \sigma_b^2 \\
&u_1 = \sqrt{q_h^{l-1}(x_1)} z_1, \quad u_2 = \sqrt{q_h^{l-1}(x_2)} \left[c_h^{l-1}(x_1,x_2) z_1 + \sqrt{1-(c_h^{l-1}(x_1,x_2))^2} z_2\right].
\end{aligned}
\tag{2.4}
$$

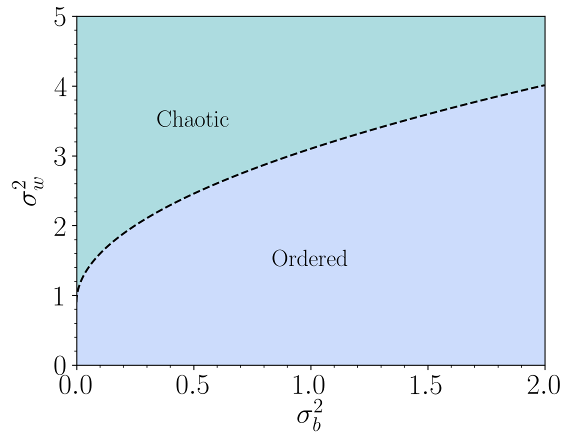
Here $c_h^l(x_1,x_2) = q_h^l(x_1,x_2)/\sqrt{q_h^l(x_1)q_h^l(x_2)}$ is the correlation coefficient between the two signals reaching layer l, and Dz is the standard Gaussian measure.

Ref. [38] found that the signal's length reaches its fixed point within very few layers, and the fixed point of the correlation coefficient, $c_h^*(x_1,x_2)$, can be estimated with the assumption that $q_h^l(x)$ has reached its fixed point $q^*(x)$. We can write Eqn. 2.4 under this assumption

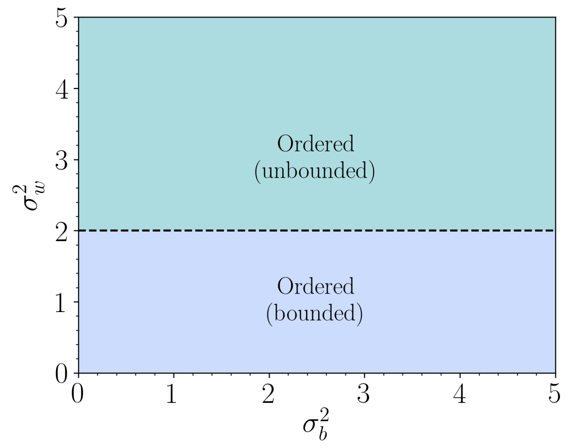
$$\begin{aligned}
&\mathscr{C}\left(c_h^{l-1}(x_1,x_2), q_h^*(x) \mid \sigma_w^2, \sigma_b^2\right) = \sigma_w^2 \int Dz_1 Dz_2\, \phi(u_1)\phi(u_2) + \sigma_b^2 \\
&u_1 = \sqrt{q_h^*(x)}\, z_1, \quad u_2 = \sqrt{q_h^*(x)}\left[c_h^{l-1}(x_1,x_2)\, z_1 + \sqrt{1-(c_h^{l-1}(x_1,x_2))^2}\, z_2\right].
\end{aligned} \tag{2.5}$$

It is easy to see that $c_h^*(x_1,x_2) = 1$ is always a fixed point of the recursive map. The stability of the fixed point $c_h^*(x_1,x_2) = 1$ is determined by

$$\chi_1 \equiv \frac{\partial c_h^l(x_1,x_2)}{\partial c_h^{l-1}(x_1,x_2)}\Big|_{c_h^{l-1}(x_1,x_2)=1} = \sigma_w^2 \int Dz\, [\phi'(\sqrt{q^*(x)}\, z)]^2.$$



(a)



(b)

Figure 2.1: Phase diagram for tanh and ReLU networks with Gaussian distributed weights. (a) tanh networks have two phases. First, an 'ordered' phase where the correlations converge to one, and second a 'chaotic' phase, where the correlations converge to fixed point $c_h^*(x_1,x_2) < 1$. The two phases are separated by the boundary $\chi = 1$. (b) ReLU networks with uncorrelated weights also have two phases. First, a 'bounded' phase where length's fixed point $q^*(x)$ is finite, and second, an 'unbounded' phase in which the length diverges. The two phases are separated by $\sigma_w^2 = 2$. In both phases, any two signals will eventually become correlated.

χ_1 separates the parameter space into two phases - first, an 'ordered' phase with $\chi_1 < 1$, where the $c_h^*(x_1,x_2) = 1$ fixed point is stable; and second, a 'chaotic' phase with $\chi_1 > 1$, where the $c_h^*(x_1,x_2) = 1$ fixed point is unstable. $\chi_1 = 1$ defines the phase boundary line. Figure 2.1a shows the phase diagram of tanh activation. In the ordered phase, two distinct signals will become perfectly correlated asymptotically. In the chaotic phase, the correlations converge to a stable fixed point below unity. In this phase, even two closely related signals will eventually lose correlations. Both the ordered and the chaotic phase modify the correlations between the input signals. If this modification is too drastic, it might lead to training failure. For example, if we consider a cat vs. dog image classification task. The network may find it difficult to differentiate between a cat and a dog image in the ordered phase, whereas it may face difficulties in keeping two cat images in the same class in a chaotic phase. The parameters at the phase transition boundary do not modify the input correlations, and therefore, correspond to optimal information flow through the network. Note that these problems would only occur in deep neural networks; shallow networks do not suffer from this problem.

Ref. [40] extended this analysis by deriving the depth scales associated with the fixed points.

We begin by expanding the recurrence relations around the fixed point up to the first order of magnitude, i.e.,

$$\begin{aligned} q_h^l(x) &= q^*(x) + \varepsilon_q^l \\ c_h^l(x_1, x_2) &= 1 + \varepsilon_c^l. \end{aligned}$$

Here, we have already assumed that the fixed points exist, which implies $\varepsilon_q^l, \varepsilon_c^l \to 0$ as $l \to \infty$ .This might not be valid in general (for example, see the analysis for the ReLU function below). Furthermore, we write $\varepsilon_q^l \sim e^{-l/\xi_q}$, $\varepsilon_c^l \sim e^{-l/\xi_c}$, where ξ_q, ξ_c are the depth scales given by

$$\begin{aligned} \xi_q^{-1} &= -\log\left[\chi_1 + \sigma_w^2 \int Dz\, \phi''(\sqrt{q^*(x)}z)\phi(\sqrt{q^*(x)}z)\right] \\ \xi_c^{-1} &= -\log\left[\sigma_w^2 \int Dz_1\, Dz_2\, \phi'(u_1^*)\phi(u_2^*)\right] \\ u_1^* &= \sqrt{q^*(x)}z_1; \sqrt{q^*(x)}\left(c^*(x_1,x_2)\, z_1 + \sqrt{1-(c^*(x_1,x_2))^2\, z_2}\right). \end{aligned}$$

The correlation coefficient depth scale further simplifies to $\xi_c^{-1} = -\log \chi_1$ in the ordered phase $c^*(x_1, x_2) = 1$. It is easy to see that the depth scale ξ_c^{-1} diverges at the phase transition boundary given by $\chi_1 = 1$, given that $q^*(x), c^*(x_1, x_2)$ exist.

The discussion so far in this section applies to a general non-linear activation ϕ. Now we will focus on the case of ReLU activation. The above analysis is applied assuming $q^*(x)$ is finite shows that ReLU networks do not have a chaotic phase, and any two signals propagating through a ReLU network become asymptotically correlated for all values of (σ_w^2, σ_b^2). In other words, $c^*(x_1, x_2) = 1$ is always a stable fixed point in the regime where signal strength is bounded.

However, we can still classify the parameter space of the network into two phases based on the boundedness of the fixed point $q^*(x)$ of the length map (Eqn. 2.3). First, a 'bounded' phase where $q^*(x)$ is finite and non-zero; second, an 'unbounded' phase, where $q^*(x)$ is either zero or infinite [41, 53]. Fig. 2.1b depicts the phase diagram for ReLU networks. The two phases are separated by the boundary $\sigma_w^2 = 2$. Initializing ReLU networks with $(\sigma_w^2, \sigma_b^2) = (2, 0)$ is commonly known as He initialization [3]. The analysis of the stability of $c^*(x_1,x_2) = 1$ fixed point in ReLU networks is valid only in the bounded phase. However, numerical results presented below indicate that the fixed point remains stable even in the unbounded phase.

In Fig. 2.2, we show the values of the length and correlation coefficient as a function of the variance of weights after uncorrelated signals have propagated through l layers. The above quantities were calculated by first averaging over $M = 1024$ input signals, then averaging over 40 networks initialized with independently picked $w^l_{ij} \sim \mathcal{N}(0, \sigma_w^2/N)$, where we have considered a constant width $N_l = N = 2048$. As the critical boundaries do not depend on the variance of bias, we show results for $\sigma_b^2 = 0.1$ only. We observe the results as predicted by Eqn. 2.2. The signal's length starts to diverge at $\sigma_w^2 = 2$, and the correlation coefficient after propagation through 32 layers is almost one. Therefore, the numerical results validate the predicted results, validating the mean-field analysis.

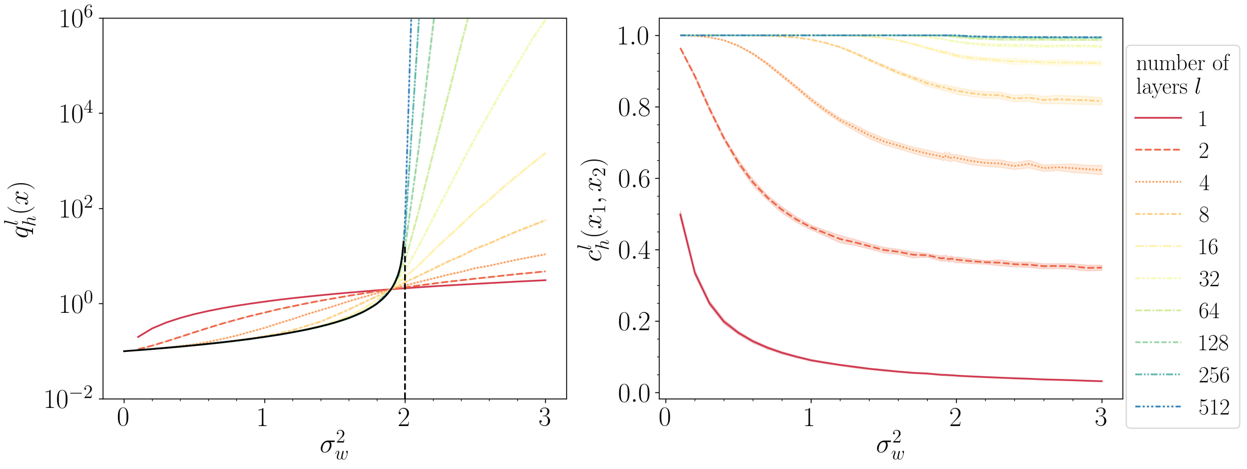


Figure 2.2: The above plots show the length and the correlation coefficient of the signals after propagating through l layers in a ReLU network. We calculate the length and the correlation coefficient for $M = 1024$ input signals, averaged over 40 networks with a constant width $N = 2048$. In the first panel, the dashed line indicates the theoretical phase boundary $\sigma_w^2 = 2$, and the solid black line denotes the theoretical prediction for the length's fixed point. The theoretical results correctly predict the length and correlation coefficient's fixed point, validating the mean-field analysis. Note that the apparent crossing point before $\sigma_w^2 = 2$ is due to finite bias, and the phase boundary is indeed at $\sigma_w^2 = 2$. On considering zero bias, we observe that the crossing point overlaps with the critical point. As the critical boundaries do not depend on the variance of bias, we show results for $\sigma_b^2 = 0.1$ only.

High expressivity is a crucial feature of a deep neural network. A network with high expressive power could represent complex functions. Using Reimann geometry, Ref. [38] demonstrated that deep neural networks achieve exponential expressivity in a chaotic phase as a function of depth. However, as correlations between two signals always converge to one, ReLU networks may not be very expressive. Multiple studies suggest that ReLU networks are biased towards simpler functions, supporting the observation. If that is the case, then ReLU networks should be incapable of performing complex tasks. However, they do work in practice, despite their poor expressive power. Of particular interest are two studies, which suggest that ReLU networks can be more expressive with anti-correlated weights. First, Ref. [52] found that a trained ReLU CNN's feature matrices are negatively correlated. Another study, Ref. [43], reveals that ReLU networks with anti-correlated weights explore a richer function space on applying perturbations when compared with those with uncorrelated weights. This motivated us to analyze the critical properties of anti-correlated ReLU networks. In Section 3.1, we analyze the mean-field theory of signal propagation with anti-correlated weights. Furthermore, we show that ReLU networks with anti-correlated weights have a chaotic phase, unlike uncorrelated weights. Before that, we review an alternative method to derive the mean-field equations using generating functionals in the following section.

2.2 Mean-field theory of signal propagation using a path integral approach

This section reviews a path integral-based approach to derive mean-field equations for signal propagation in deep neural networks introduced by Ref. [43], which study perturbations applied to deep neural networks. However, we observed that the method above yields the mean-field equations for signal propagation if no perturbation is applied. The key difference from the approach in Section 2.1 is that Ref. [43] defines the covariance matrix over the activations to track the information flow instead of pre-activations. However, the equations observed are the same, up to a variable change, and the fixed point analysis remains valid. Therefore, this section unifies the two approaches by providing a relation between the averaged covariance matrix for activations and pre-activations. We also demonstrate the effect of perturbations on the fixed point $c_h^*(x_1, x_2) = 1$. Note that we do not use this methodology in the next chapter on methods as the calculations become increasingly complicated.

We consider two signals, x_1 and x_2, propagating through two deep neural networks in a variant of a teacher-student setup. To differentiate between the teacher and student network parameters, we use '^' to denote teacher variables. First, we draw the teacher network weights from a zero-mean Gaussian distribution $\hat{w}_{ij}^l \sim \mathcal{N}(0, \sigma_w^2/N_{l-1})$ and then obtain the student weights by applying a perturbation of form $w_{ij}^l = \sqrt{1-(\eta^l)^2}\hat{w}_{ij}^l + \eta^l \delta w_{ij}^l$, where η^l is the perturbation strength at layer l and $\delta w_{ij}^l \sim \mathcal{N}(0, \sigma_w^2/N_{l-1})$, drawn independently of $\hat{w}_{ij}^l$. Similarly, we draw the teacher bias from a Gaussian distribution $\hat{b}_i^l \sim \mathcal{N}(0, \sigma_b^2)$ and obtain the student bias by the same procedure $b_i^l = \sqrt{1-(\eta^l)^2}\hat{b}_i^l + \eta^l \delta b_i^l$. Note that the limit $\eta^l \to 0$ corresponds to the signal propagation problem described in the last section. We can think of this setup as a toy model to study random perturbations applied to a deep neural network while training. For example, to study randomness introduced by mini-batch and dropout procedures.

For the activation $s_i^l(x)$, consider the joint probability

$$P\left(s_i^l(x)|\boldsymbol{w}^l, \boldsymbol{s}^{l-1}(x)\right) = \sqrt{\frac{\beta}{2\pi}} \exp\left(-\frac{\beta}{2}\left[s_i^l(x) - \phi(h_i^l(x))\right]^2\right),$$

where β quantifies the strength of noise, introduced for calculation purposes. However, we can use it to study noisy computations, which is a separate problem in itself. Similar to Section 2.1, we track the layer-wise overlap of the two signals by the covariance matrix between the activations

$$q_s^l(x_a) = \frac{1}{N_l}\sum_{i=1}^{N_l}\left(s_i^l(x_a)\right)^2 \text{ where } a \in \{1,2\}$$
$$q_s^l(x_1, x_2) = \frac{1}{N_l}\sum_{i=1}^{N_l} s_i^l(x_1) s_i^l(x_2).$$

Later, we will observe that this covariance matrix is the 'natural' choice of order parameters for this system.

Consider a generating functional of form

$$\Gamma[\hat{\psi}, \psi] = \left\langle \exp(-\imath \sum_{l,i}(\hat{\psi}_i^l \hat{s}_i^l(x_1) + \psi_i^l s_i^l(x_2)) \right\rangle,$$

where the angular brackets $\langle .. \rangle$ represent an average over the weights, biases, and the internal degrees of freedom (activations and pre-activations). We can obtain the quantities of interest by calculating appropriate derivatives of the generating functional. For example,

$$\left\langle q_s^l(x_1,x_2)\right\rangle = -\frac{1}{N_l}\sum_{i=1}^{N_l}\lim_{\hat{\psi}_i^l,\psi_i^l\to 0}\frac{\partial^2}{\partial\hat{\psi}_i^l\partial\psi_i^l}\left\langle \mathbf{\Gamma}[\hat{\psi},\psi]\right\rangle .$$

For the ease of future notations, we will denote the averaged generating functional by $\mathbf{\Gamma}$, the averaged overlap by q^l, and the activations and pre-activations by s_i^l and h_i^l. For simplicity, we will consider a constant width across all layers $N_l = N$, but the results are valid for all N_l, as long as it is large. We aim to write the averaged generating functional as $\mathbf{\Gamma}\sim e^{N\Psi}$, where Ψ is called the rate function. In the infinite width limit $N\to\infty$, the generating functional is dominated by the extremum of Ψ. Therefore, the condition for the extremum of Ψ will give us the mean-field conditions.

The averaged generating functional has the form

$$\begin{aligned}\Gamma = \int\prod_{l,i,j=1}^{L,N} d\hat{\boldsymbol{w}}_{ij}^l\, d(\delta\boldsymbol{w}_{ij}^l)\, d\hat{\boldsymbol{b}}_i^l\, d(\delta\boldsymbol{b}_i^l)\, P(\hat{\boldsymbol{w}}_{ij}^l)P(\delta\boldsymbol{w}_{ij}^l)P(\hat{b}_i^l)P(\delta b_i^l)\times \\ \times\int\prod_{l=0,i=1}^{L,N} d\hat{s}_i^l\, ds_i^l P\left(\hat{s}_i^l|\hat{\boldsymbol{w}}^l,\hat{\boldsymbol{s}}^{l-1}\right)P\left(s_i^l|\boldsymbol{w}^l,\boldsymbol{s}^{l-1}\right)\exp\left(-\imath(\hat{\psi}_i^l\hat{s}_i^l+\psi_i^l s_i^l)\right)\end{aligned} \tag{2.6}$$

We will introduce the relation $h_i^l = \sum_j w_{ij}^l s_j^{l-1}$ through the integral representation of the δ function

$$\begin{aligned}\delta(\hat{h}_i^l - \sum_j \hat{w}_{ij}^l\hat{s}_j^{l-1}) &= \int\frac{d\hat{x}_i^l}{2\pi}e^{\imath\hat{x}_i^l\left(\hat{h}_i^l-\sum_j\hat{w}_{ij}^l\hat{s}_j^{l-1}\right)}\\ \delta(h_i^l - \sum_j w_{ij}^l s_j^{l-1}) &= \int\frac{dx_i^l}{2\pi}e^{\imath x_i^l\left(h_i^l-\sum_j w_{ij}^l s_j^{l-1}\right)}.\end{aligned}$$

Introducing the above equations makes the generating functional representation complicated. So, we focus on the weights and activation integrals separately. First, we will consider the weight average, which gives

$$= \prod_{l,i,j=1}^{L,N} \int d\hat{w}_{ij}^l dw_{ij}^l P(\hat{w}_{ij}^l) P(\delta w_{ij}^l) \exp\left(-\iota \left[\hat{w}_{ij}^l \left(\hat{x}_i^l \hat{s}_j^{l-1} + \sqrt{1-(\eta^l)^2}\, x_i^l s_j^{l-1}\right) + \eta^l \delta w_{ij}^l x_i^l s_j^{l-1}\right]\right)$$
$$= \prod_{l,i=1}^{L,N} \exp\left(-\frac{\sigma_w^2}{2}\left[(\hat{x}_i^l)^2 \left(\frac{\sum_j (\hat{s}_j^l)^2}{N}\right) + (x_i^l)^2 \left(\frac{\sum_j (s_j^l)^2}{N}\right) + 2\sqrt{1-(\eta^l)^2}\hat{x}_i^l x_i^l \left(\frac{\sum_j \hat{s}_j^l s_j^l}{N}\right)\right]\right).$$

If we introduce the following order parameters through the integral representations of the δ function

$$\delta(q_{11}^l - \frac{1}{N}\sum_j (\hat{s}_j^l)^2) = \int \frac{dQ_{11}^l}{2\pi/N} e^{\iota N Q_{11}^l [q_{11}^l - \frac{1}{N}\sum_j (\hat{s}_j^l)^2]}$$
$$\delta(q_{22}^l - \frac{1}{N}\sum_j (s_j^l)^2) = \int \frac{dQ_{22}^l}{2\pi/N} e^{\iota N Q_{22}^l [q_{22}^l - \frac{1}{N}\sum_j (s_j^l)^2]}$$
$$\delta(q_{12}^l - \frac{1}{N}\sum_j (\hat{s}_j^l s_j^l)) = \int \frac{dQ_{12}^l}{2\pi/N} e^{\iota N Q_{12}^l [q_{12}^l - \frac{1}{N}\sum_j (\hat{s}_j^l s_j^l)]},$$

the result from the weight average simplifies to

$$= \prod_{l,i=1}^{L,N} \exp\left(-\frac{\sigma_w^2}{2}\left[(\hat{x}_i^l)^2 q_{11}^{l-1} + (x_i^l)^2 q_{22}^{l-1} + 2\sqrt{1-(\eta^l)^2}\hat{x}_i^l x_i^l q_{12}^{l-1}\right]\right).$$

We can perform the bias integral similarly, which yields

$$= \prod_{l,i=1}^{L,N} \int d\hat{b}_i^l db_i^l P(\hat{b}_i^l) P(\delta b_i^l) \exp\left(-\iota \left[\hat{b}_i^l \left(\hat{x}_i^l + \sqrt{1-(\eta^l)^2}\, x_i^l\right) + \eta^l \delta b_i^l x_i^l\right]\right)$$
$$= \prod_{l,i=1}^{L,N} \exp\left(-\frac{\sigma_b^2}{2}\left[(\hat{x}_i^l)^2 + (x_i^l)^2 + 2\sqrt{1-(\eta^l)^2}\hat{x}_i^l x_i^l\right]\right).$$

We combine the results from the weight and bias averages, which gives

$$= \prod_{l,i=1}^{L,N} \exp\left(-\frac{1}{2}\left[(\hat{x}_i^l)^2\left(\sigma_w^2 q_{11}^{l-1}+\sigma_b^2\right)+(x_i^l)^2\left(\sigma_w^2 q_{22}^{l-1}+\sigma_b^2\right)+2\sqrt{1-(\eta^l)^2}\hat{x}_i^l x_i^l\left(\sigma_w^2 q_{12}^{l-1}+\sigma_b^2\right)\right]\right).$$

Next, we introduce the doublet $X_i^l = \begin{bmatrix}\hat{x}_i^l & x_i^l\end{bmatrix}^T$ to write the above equation compactly as

$$= \prod_{l,i} \exp\left(-\frac{1}{2}(X_i^l)^T \Sigma_{l-1} X_i^l\right),$$

where the covariance matrix Σ_{l-1} is given by

$$\Sigma_{l-1} = \begin{bmatrix} q_{11}^{l-1} & \sqrt{1-(\eta^l)^2}q_{12}^{l-1} \\ \sqrt{1-(\eta^l)^2}q_{12}^{l-1} & q_{22}^{l-1}\end{bmatrix}$$

Finally , we substitute this result to get the disordered averaged (weights and bias) generating functional

$$\begin{aligned}\Gamma = \prod_{l=0}^{L}\int \frac{dQ_{11}^l dq_{11}^l}{2\pi/N}\frac{dQ_{22}^l dq_{22}^l}{2\pi/N}\frac{dQ_{12}^l dq_{12}^l}{2\pi/N} e^{\iota N\left[Q_{11}^l q_{11}^l + Q_{22}^l q_{22}^l + Q_{12}^l q_{12}^l\right]} \times \\ \times \prod_{l,i=1}^{L,N}\int \frac{d\hat{h}_i^l d\hat{x}_i^l}{2\pi}\frac{dh_i^l dx_i^l}{2\pi}\int d\hat{s}_i^l\, ds_i^l\, e^{\iota (X_i^l)^T H_i^l}\, e^{-\frac{1}{2}(X_i^l)^T\Sigma_{l-1}X_i^l} \times \\ \times \prod_{l=0,i=1}^{L,N} e^{-\iota\left[Q_{11}^l(\hat{s}_i^l)^2 + Q_{22}^l(s_i^l)^2 + Q_{12}^l\hat{s}_i^l s_i^l\right]} e^{-\iota(\hat{\psi}_i^l\hat{s}_i^l + \psi_i^l s_i^l)} \times \\ \times \prod_{l,i=1}^{L,N} P\left(\hat{s}_i^l|\hat{h}_i^l\right)P\left(s_i^l|h_i^l\right),\end{aligned}$$

where we have introduced the pre-activation doublet $H_i^l = \begin{bmatrix}\hat{h}_i^l & h_i^l\end{bmatrix}^T$. Next, we perform the integral for X_i^l to simplify

$$
\begin{aligned}
\Gamma = \prod_{l=0}^{L} \int \frac{dQ_{11}^l dq_{11}^l}{2\pi/N} \frac{dQ_{22}^l dq_{22}^l}{2\pi/N} \frac{dQ_{12}^l dq_{12}^l}{2\pi/N} e^{\imath N\left[Q_{11}^l q_{11}^l + Q_{22}^l q_{22}^l + Q_{12}^l q_{12}^l\right]} \times \\
\times \prod_{l,i=1}^{L,N} \int d\hat{h}_i^l dh_i^l \int d\hat{s}_i^l \, ds_i^l \, e^{-\frac{1}{2}(H_i^l)^T \Sigma_{l-1}^{-1} H_i^l} \times \\
\times \prod_{l=0,i}^{L,N_l} e^{-\imath\left[Q_{11}^l (\hat{s}_i^l)^2 + Q_{22}^l (s_i^l)^2 + Q_{12}^l \hat{s}_i^l s_i^l + \hat{\psi}_i^l \hat{s}_i^l + \psi_i^l s_i^l\right]} \times \\
\times \prod_{l,i=1}^{L,N} P\left(\hat{s}_i^l | \hat{h}_i^l\right) P\left(s_i^l | h_i^l\right).
\end{aligned}
$$

Now, the activations and pre-activations are the same for every neuron in each layer. Therefore, we can re-write the above expression in a site-independent form

$$
\begin{aligned}
\Gamma = \prod_{l=0}^{L} \int \frac{dQ_{11}^l dq_{11}^l}{2\pi/N} \frac{dQ_{22}^l dq_{22}^l}{2\pi/N} \frac{dQ_{12}^l dq_{12}^l}{2\pi/N} e^{\imath N\left[Q_{11}^l q_{11}^l + Q_{22}^l q_{22}^l + Q_{12}^l q_{12}^l\right]} \times \\
\times \left\{ \prod_{l=1}^{L} \int d\hat{h}^l dh^l \int d\hat{s}^l \, ds^l \, e^{-\frac{1}{2}(H^l)^T \Sigma_{l-1}^{-1} H^l} \times \right. \\
\left. \times \prod_{l=0}^{L} e^{-\imath\left[Q_{11}^l (\hat{s}_i^l)^2 + Q_{22}^l (s_i^l)^2 + Q_{12}^l \hat{s}_i^l s_i^l + \hat{\psi}^l \hat{s}^l + \psi^l s^l\right]} \right\}^N \times \\
\times \prod_{l=1}^{L} P\left(\hat{s}^l | \hat{h}_i^l\right) P\left(s^l | h_i^l\right).
\end{aligned}
$$

We re-write the above expression as

$$
\Gamma = \prod_{l=0}^{L} \int \frac{dQ_{11}^l dq_{11}^l}{2\pi/N} \frac{dQ_{22}^l dq_{22}^l}{2\pi/N} \frac{dQ_{12}^l dq_{12}^l}{2\pi/N} \, e^{N\Psi\left[q_{11}^l, Q_{11}^l, q_{22}^l, Q_{22}^l, q_{12}^l, Q_{12}^l\right]},
$$

where the rate function Ψ is given by

$$\Psi[q_{11}^l, Q_{11}^l, q_{22}^l, Q_{22}^l, q_{12}^l, Q_{12}^l] = \imath \sum_{l=0}^{L} \left(Q_{11}^l q_{11}^l + Q_{22}^l q_{22}^l + Q_{12}^l q_{12}^l \right) + \log \int d\hat{h}^l dh^l \prod_{l=1}^{L} M[\hat{s}^l, s^l, \hat{h}^l, h^l],$$

where $M[...]$ has the form

$$M[\hat{s}^l, s^l, \hat{h}^l, h^l] = \prod_{l=0}^{L} e^{-\imath\left[Q_{11}^l (\hat{s}^l)^2 + Q_{22}^l (s^l)^2 + Q_{12}^l (\hat{s}^l s^l) + \hat{\psi}^l \hat{s}^l + \psi^l s^l\right]} \times$$
$$\times \prod_{l=1}^{L} P\left(\hat{s}^l | \hat{h}^l\right) P\left(s^l | h^l\right) e^{-\frac{1}{2}(H^l)^T \Sigma_{l-1}^{-1} H^l}.$$

In the limit $N \to \infty$, Γ is dominated by the saddle point of $\Psi[...]$, obtained by setting $\frac{\partial \Psi}{\partial q^l} = \frac{\partial \Psi}{\partial Q^l} = 0$ for both length and overlap. The conjugate variables vanish at the saddle point

$$Q_1^l = Q_{22}^l = Q_{12}^l = 0 \quad \forall l.$$

This yields the mean-field equations

$$\begin{aligned}
q_{11}^l &= \frac{1}{\beta} + \int d\hat{h}^l dh^l \left(\phi(\hat{h}^l)\right)^2 \frac{e^{-\frac{1}{2}(H^l)^T \Sigma_{l-1}^{-1} H^l}}{\sqrt{(2\pi)^2 |\Sigma_{l-1}|}} \\
q_{22}^l &= \frac{1}{\beta} + \int d\hat{h}^l dh^l \left(\phi(h^l)\right)^2 \frac{e^{-\frac{1}{2}(H^l)^T \Sigma_{l-1}^{-1} H^l}}{\sqrt{(2\pi)^2 |\Sigma_{l-1}|}} \\
q_{12}^l &= \int d\hat{h}^l dh^l \phi(\hat{h}^l) \phi(h^l) \frac{e^{-\frac{1}{2}(H^l)^T \Sigma_{l-1}^{-1} H^l}}{\sqrt{(2\pi)^2 |\Sigma_{l-1}|}}.
\end{aligned}$$

The above equation describes how the signal propagates through a deep neural network when a perturbation of η^l is applied at layer l. Next, we consider the noiseless limit $\beta \to \infty$

$$
\begin{aligned}
q_{11}^l &= \int d\hat{h}^l dh^l \left(\phi(\hat{h}^l)\right)^2 \frac{e^{-\frac{1}{2}(H^l)^T \Sigma_{l-1}^{-1} H^l}}{\sqrt{(2\pi)^2 |\Sigma_{l-1}|}} \\
q_{22}^l &= \int d\hat{h}^l dh^l \left(\phi(h^l)\right)^2 \frac{e^{-\frac{1}{2}(H^l)^T \Sigma_{l-1}^{-1} H^l}}{\sqrt{(2\pi)^2 |\Sigma_{l-1}|}} \\
q_{12}^l &= \int d\hat{h}^l dh^l \phi(\hat{h}^l)\phi(h^l) \frac{e^{-\frac{1}{2}(H^l)^T \Sigma_{l-1}^{-1} H^l}}{\sqrt{(2\pi)^2 |\Sigma_{l-1}|}}.
\end{aligned}
$$

We can re-write the above equations in the form that resembles Eqn. 2.2. The length map $\hat{\mathscr{V}}$ is given by

$$
q_s^l(x) = \hat{\mathscr{V}}\left(q_s^{l-1}(x) \mid \sigma_w^2, \sigma_b^2, \eta^l\right) = \int Dz\, \phi\left(\sqrt{\sigma_w^2 q_s^{l-1}(x) + \sigma_b^2}\, z\right)^2,
$$

and the correlation map $\hat{\mathscr{C}}$ under the assumption $q_{11}^l, q_{22}^l \to q^*$ is given by

$$
\begin{aligned}
&q_s^l(x_1,x_2) = \hat{\mathscr{C}}\left(c_s^{l-1}(x_1,x_2), q_s^{l-1}(x_1), q_s^{l-1}(x_2) \mid \sigma_w^2, \sigma_b^2, \eta^l\right) = \int Dz_1 Dz_2\, \phi(u_1)\phi(u_2) \\
&u_1 = \sqrt{\sigma_w^2 q_h^{l-1}(x_1) + \sigma_b^2}\, z_1, \quad u_2 = \sqrt{\sigma_w^2 q_h^{l-1}(x_2) + \sigma_b^2}\left[\hat{c}_s^{l-1}(x_1,x_2) z_1 + \sqrt{1-(\hat{c}_s^{l-1}(x_1,x_2))^2}\, z_2\right].
\end{aligned}
$$

Here $\hat{c}_s^{l-1}(x_1,x_2)$ is

$$
\hat{c}_s^{l-1}(x_1,x_2) = \sqrt{1-(\eta^l)^2}\, \frac{\sigma_w^2 q_{12}^l + \sigma_b^2}{\sigma_w^2 q^* + \sigma_b^2}
$$

We observe that $c_s^*(x_1,x_2) = 1$ is not the fixed point of the recursive map unless $\eta^l = 0$. Therefore, perturbations applied to a deep neural network destroy the fixed point $c_s^*(x_1,x_2) = 1$. Note that similar results are observed by Ref. [40]. It is easy to see that the limit $\eta \to 0$ gives Eqn. 2.2, relating the covariance matrix of the activations to the pre-activations

$$\Sigma_l = \begin{bmatrix} \sigma_w^2 q_s^l(x_1) + \sigma_b^2 & \sigma_w^2 q_s^l(x_1,x_2) + \sigma_b^2 \\ \sigma_w^2 q_s^l(x_1,x_2) + \sigma_b^2 & \sigma_w^2 q_s^l(x_2) + \sigma_b^2 \end{bmatrix} = \begin{bmatrix} q_h^l(x_1) & q_h^l(x_1,x_2) \\ q_h^l(x_1,x_2) & q_h^l(x_2) \end{bmatrix}.$$

2.3 Training tasks

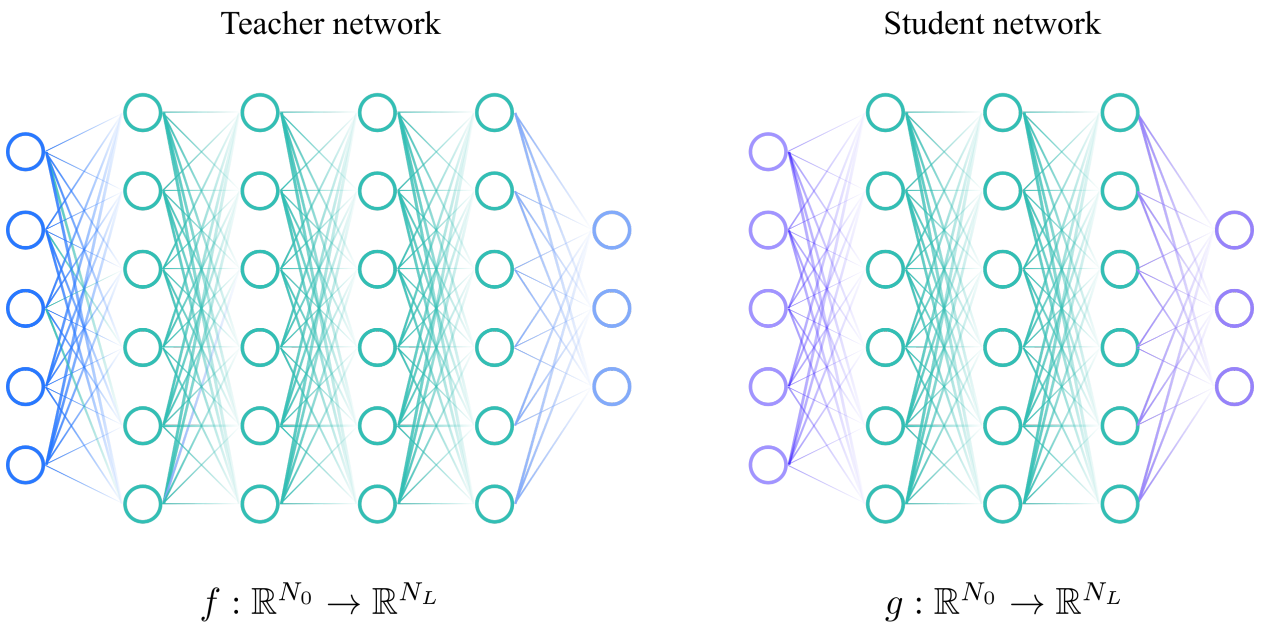

Teacher network
Student network
$f : \mathbb{R}^{N_0} \to \mathbb{R}^{N_L}$
$g : \mathbb{R}^{N_0} \to \mathbb{R}^{N_L}$

Figure 2.3: Teacher-student setup: An untrained teacher network $f : \mathbb{R}^{N_0} \to \mathbb{R}^{N_L}$ is used to generate a training set $\mathscr{T} = \{x, f(x)\}$, where $x \in \mathscr{N}(0,1)$. A student network $g : \mathbb{R}^{N_0} \to \mathbb{R}^{N_L}$ is trained over the training set $\mathscr{T}$ to minimize the difference $|f - g|$.

This section describes various tasks used to analyze the training dynamics and performance of deep neural networks with different initialization schemes. We consider two types of tasks —randomly generated functions and real datasets.

First, we consider a variant of teacher-student setup [54], which helps understand the training dynamics without dealing with the complexities of the data structure in real datasets, and allows us to average over the training dataset. We consider a teacher neural network $f : \mathbb{R}^{N_0} \to \mathbb{R}^{N_L}$ with untrained fixed parameters to generate a training dataset $\mathscr{T} = \{x, f(x)\}$, where the input signals x are drawn from standard normal distribution $\mathscr{N}(0,1)$. Next, we train a student network (not necessarily of the same size) over this training dataset to approximate this teacher function f. Let's denote this approximation by $g : \mathbb{R}^{N_0} \to \mathbb{R}^{N_L}$. The student network parameters are optimized to minimize the difference between the teacher and student functions $|f - g|$. This procedure is pictorially shown in Figure 2.3.

We consider an $L = 10$ layered student neural network with $N = 100$ neurons in each layer with a mean-squared loss and SGD/Adam optimizer, trained by data generated by teacher networks described below using 10^5 input signals drawn from the standard normal distribution. We consider three different tasks with varying complexities.

1. A standard teacher task, in which the training data is generated by a ReLU network of the same size as the student network, initialized using He initialization.

2. Next, we consider a simple teacher network with capacity (size) much lower than the student network. In many real data sets, the high-dimensional inputs lie in a low-dimensional manifold [55], indicating low complexity of data. This motivates us to consider an easy teacher task. We consider a single-layer ReLU network with only $N = 10$ neurons, initialized with He initialization.

3. Lastly, we consider a complex teacher task, in which the complexity of the teacher network is more than the student network. We consider a teacher network with tanh activation of the same size as the student initialized in the chaotic regime [38], $(\sigma_w^2, \sigma_b^2) = (1.5, 0)$ (For ReLU networks initialized with symmetric distributions, half of the neurons are dead. Thus, the effective capacity of a ReLU network is half its size).

The various teacher tasks are summarized in Table 2.1. We implement neural networks in Tensorflow (version 2.2.0) [56] and train them with a mean squared error loss with a mini-batch size of 1000 and default parameters for the optimizers.

Task	size of teacher network	activation	(σ_w^2, σ_b^2)
standard task	same as student network	ReLU	(2, 0)
simple task	single hidden layer with 10 neurons	ReLU	(2, 0)
complex task	same as student network	tanh	(1.5, 0)

Table 2.1: Three tasks used to probe the training dynamics and performance of different initialization schemes.

Lastly, we also verify our results on two image classification datasets. First, the MNIST dataset, it contains $|\mathscr{T}| = 60,000$ handwritten digit images for training and $10,000$ for testing. It is an overused and straightforward training task, for which we can quickly achieve 98% classification accuracy. Therefore, we also consider the Fashion MNIST dataset, which contains $|\mathscr{T}| = 60,000$ fashionwear images for training and $10,000$ for testing. It is a slightly complex task, for which we can achieve reasonable accuracy even with feed-forward neural networks. For the real datasets, we train neural networks with categorial cross entropy loss with a mini-batch size of 600. This choice of mini-batch size ensures that the number of steps in each epoch is the same for the teacher-student and image classification tasks.

We consider an $L = 10$ layered neural network with $N = 100$ neurons in each layer (except first and last layer which are restricted by input and output size) with a cross entropy loss

$$\text{Cross entropy} = -\frac{1}{|\mathscr{T}|}\sum_{i\in\mathscr{T}}^{|\mathscr{T}|}\sum_{c}^{classes} \hat{s}_{i,c}^{L}\, log(f(s_{i,c}^{L})),$$

where $\hat{s}_{i,c}^{L}$ $f(s_{i,c}^{L})$ are the desired and output probability for the class c and input i. We train the networks using SGD and Adam optimizer, and validate our output using the classification accuracy metric

$$\text{Classification accuracy} = \frac{\text{Number of correct predictions}}{\text{Total number of predictions}}.$$

Chapter 3

Results

3.1 Mean-field analysis of signal propagation with correlated weights

In this section, we provide a mean-field analysis complementary to that of ReLU networks with anti-correlated weights. Unlike Ref. [43, 44], which studies perturbation around a ReLU network, our analysis aims to understand the critical properties of correlations between input signals. We consider correlations within the set of weights ($\boldsymbol{w}_i^l$) incoming to each neuron i, with all neurons identically distributed. The correlated Gaussian distribution is

$$P(\boldsymbol{w}_1^l, \boldsymbol{w}_2^l, \boldsymbol{w}_3^l \dots) = \prod_i^{N_l} \frac{\exp\left(-\frac{1}{2}(\boldsymbol{w}_i^l)^T A^{-1} \boldsymbol{w}_i^l\right)}{\sqrt{(2\pi)^{N_{l-1}}|A|}}, \tag{3.1}$$

with a covariance matrix given by

$$A = \frac{\sigma_w^2}{N_{l-1}} \left(\mathbb{I} - \frac{k}{1+k} \frac{J}{N_{l-1}} \right).$$

Here $\mathbb{I}$ is the identity matrix, J is an all-ones matrix, and k parameterizes the correlation strength. Positively correlated and anti-correlated regimes correspond to the regions $-1 < k < 0$ and $k > 0$, respectively, which correspond to the off-diagonal elements of A being positive and negative. Note that weights reaching two different nodes are uncorrelated, and also the bias is uncorrelated with the weights.

We follow the same approach discussed in Section 2.1 to derive the length and correlation maps for ReLU networks with correlated weights given by Eqn. 3.1. Assuming self-averaging, we obtain the average value of the length of a signal, x, reaching layer l by considering an average over weights and biases as

$$\begin{aligned} q_h^l(x) &= \frac{1}{N_l}\left\langle \sum_{i=1}^{N_l} (h_i^l(x))^2 \right\rangle = \frac{1}{N_l}\sum_{i=1}^{N_l}\sum_{j,m=1}^{N_{l-1}} \left\langle w_{ij}^l w_{im}^l \right\rangle \phi(h_j^{l-1}(x))\phi(h_m^{l-1}(x)) + \left\langle (b_i^l)^2 \right\rangle \\ &= \frac{\sigma_w^2}{N_{l-1}} \sum_{j,m=1}^{N_{l-1}} \left(\delta_{j,m} - \frac{k}{1+k}\frac{1}{N_{l-1}}\right) \phi(h_j^{l-1}(x))\phi(h_m^{l-1}(x)) + \sigma_b^2, \end{aligned}$$

where we have used $\left\langle w_{ij}^l w_{im}^l \right\rangle = \frac{\sigma_w^2}{N_{l-1}}\left(\delta_{j,m} - \frac{k}{1+k}\frac{1}{N_{l-1}}\right)$ and $\langle (b_i^l)^2 \rangle = \sigma_b^2$. For large N_{l-2}, each $h_i^{l-1}(x)$ is a weighted sum of a large number of correlated random variables, which converges to a zero-mean Gaussian with a variance $q_h^{l-1}(x)$ (by the definition of $q_h^{l-1}(x)$). Replacing the average over all neurons at layer $l-1$ by a Gaussian distribution, we get

$$q_h^l(x) = \sigma_w^2 \int Dz\, \phi\left(\sqrt{q_h^{l-1}(x)}\, z\right)^2 - \sigma_w^2 \frac{k}{1+k}\left[\int Dz\, \phi\left(\sqrt{q_h^{l-1}(x)}\, z\right)\right]^2 + \sigma_b^2. \tag{3.2}$$

For the second term in Eqn. 3.2, we have used the fact that for $m \neq j$, $h_j^l(x)$ and $h_m^l(x)$ are uncorrelated random variables and ignored $\mathscr{O}(1/N)$ terms. Note that weights reaching two different nodes are uncorrelated. We can write Eqn. 3.2 in terms of the length map $\mathscr{V}$ from Eqn. 2.2 to get

$$q_h^l(x) = \mathscr{V}(q_h^{l-1}(x)|\sigma_w^2, \sigma_b^2) - \sigma_w^2 \frac{k}{1+k}\left[\int Dz\, \phi\left(\sqrt{q_h^{l-1}(x)}\, z\right)\right]^2.$$

For a ReLU activation, we can perform the integrals to get the exact form of the recursive relation between $q_h^l(x)$ and $q_h^{l-1}(x)$ to be

$$q_h^l(x) = \frac{\sigma_w^2 q_h^{l-1}(x)}{2}\left(1 - \frac{k}{1+k}\frac{1}{\pi}\right) + \sigma_b^2. \tag{3.3}$$

The length is bounded if $\sigma_w^2 < 2/\left(1 - \frac{k}{1+k}\frac{1}{\pi}\right)$. Thus, the boundary separating the 'bounded' and 'unbounded' phase shifts depending on the correlation strength. The boundary moves downward from that in the $k = 0$ case (see Fig. 2.1b) when the weights are positively correlated. In contrast, the boundary shifts upwards for anti-correlated weights.

The covariance map can be derived similarly by considering an average over the weights and biases

$$q_h^l(x_1,x_2) = \frac{1}{N_l}\left\langle \sum_{i=1}^{N_l} h_i^l(x_1)h_i^l(x_2)\right\rangle = \sum_{j,m=1}^{N_{l-1}}\left\langle w_{ij}^l w_{im}^l\right\rangle \phi(h_j^{l-1}(x_1))\phi(h_m^{l-1}(x_2)) + \left\langle (b_i^l)^2\right\rangle$$
$$q_h^l(x_1,x_2) = \frac{\sigma_w^2}{N_{l-1}}\sum_{j,m=1}^{N_{l-1}}\left(\delta_{j,m} - \frac{k}{1+k}\frac{1}{N_{l-1}}\right)\phi(h_j^{l-1}(x_1))\phi(h_m^{l-1}(x_2)) + \sigma_b^2,$$

and then replacing the sum over neurons in the previous layer with an integral with a Gaussian measure. For large N_{l-2}, the joint distribution of $h_j^{l-1}(x_1)$ and $h_k^{l-1}(x_2)$ will converge to a two-dimensional Gaussian distribution with a covariance $q_h^{l-1}(x_1,x_2)$. Propagating this joint distribution across one layer, we obtain the iterative map

$$q_h^l(x_1,x_2) = \sigma_w^2\int Dz_1Dz_2\phi(u_1)\phi(u_2) - \sigma_w^2\frac{k}{k+1}\int Dz_1Dz_2\ \phi(\sqrt{q_h^{l-1}(x_1)}z_1)\phi(\sqrt{q_h^{l-1}(x_2)}z_2) + \sigma_b^2$$
$$u_1 = \sqrt{q_h^{l-1}(x_1)}z_1,\ \ u_2 = \sqrt{q_h^{l-1}(x_2)}\left[c_h^{l-1}(x_1,x_2)\,z_1 + \sqrt{1-(c_h^{l-1}(x_1,x_2))^2}\,z_2\right], \tag{3.4}$$

where $c_h^l(x_1,x_2) = q_h^l(x_1,x_2)/\sqrt{q_h^l(x_1)q_h^l(x_2)}$ is the correlation coefficient, and Dz_1, Dz_2 are standard normal Gaussian distributions. Again, in the second part of Eqn. 3.4, we have used the fact that for $k \neq j$, $h_j^l(x_1)$ and $h_k^l(x_2)$ are uncorrelated random variables and have ignored $\mathscr{O}(1/N)$ terms. We can further write Eqn. 3.4 in terms of the correlation map $\mathscr{C}$ from Eqn. 2.2 to get

$$q_h^l(x_1,x_2) = \mathscr{C}\left(c_h^{l-1}(x_1,x_2), q_h^{l-1}(x_1), q_h^{l-1}(x_2) \mid \sigma_w^2, \sigma_b^2\right)$$
$$- \sigma_w^2 \frac{k}{k+1} \int Dz_1 Dz_2\, \phi(\sqrt{q_h^{l-1}(x_1)}\, z_1)\phi(\sqrt{q_h^{l-1}(x_2)}\, z_2).$$

Further, we can perform the integrals in the second part of the above equation to get

$$q_h^l(x_1,x_2) = \mathscr{C}\left(c_h^{l-1}(x_1,x_2), q_h^{l-1}(x_1), q_h^{l-1}(x_2) \mid \sigma_w^2, \sigma_b^2\right) - \frac{\sigma_w^2}{2}\frac{k}{k+1}\frac{1}{\pi}\sqrt{q_h^{l-1}(x_1) q_h^{l-1}(x_2)}.$$

As in Section 2.1, we analyze the critical properties of the correlation coefficient under the assumption that the signal length has reached a fixed point. Under this assumption, the above equation simplifies to

$$q_h^l(x_1,x_2) = \mathscr{C}\left(c_h^{l-1}(x_1,x_2), q_h^*(x) \mid \sigma_w^2, \sigma_b^2\right) - \frac{\sigma_w^2}{2}\frac{k}{k+1}\frac{q_h^*(x)}{\pi}.$$

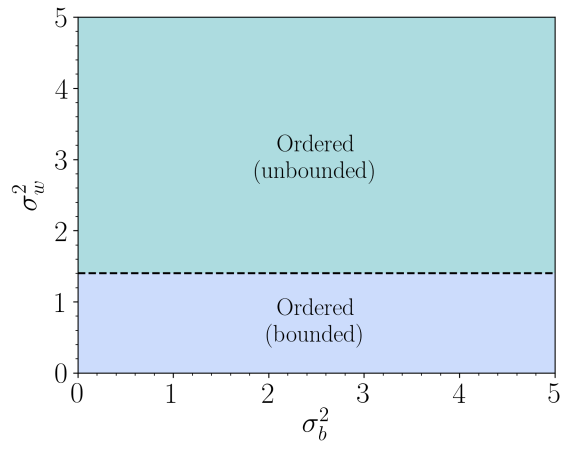
By dividing the covariance by $q_h^*(x)$, we obtain the recursive map for the correlation coefficient

$$c_h^l(x_1,x_2) = \frac{\mathscr{C}\left(c_h^{l-1}(x_1,x_2), q_h^*(x)\right)}{q_h^*(x)} - \frac{\sigma_w^2}{2}\frac{k}{k+1}\frac{1}{\pi}. \tag{3.5}$$

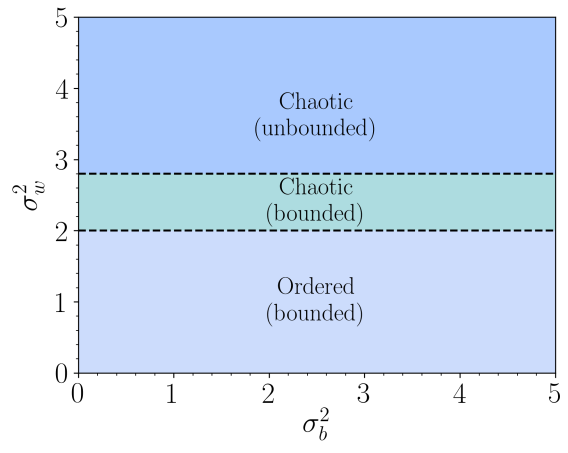
It is easy to see that $c_h^*(x_1,x_2) = 1$ is always a fixed point. Next, we determine the stability of the fixed point $c_h^*(x_1,x_2) = 1$ by calculating $\frac{\partial c_h^l(x_1,x_2)}{\partial c_h^{l-1}(x_1,x_2)}$ evaluated at the fixed point. The fixed point's stability depends only on the covariance map $\mathscr{C}$, and is given by

$$\chi_1 = \frac{\partial c_h^l(x_1,x_2)}{\partial c_h^{l-1}(x_1,x_2)}\Big|_{c_h^{l-1}(x_1,x_2)=1} = \sigma_w^2 \int Dz\, [\phi'(\sqrt{q^*(x)}\, z)]^2 = \frac{\sigma_w^2}{2}.$$

Equating the above equation to one, we obtain the phase transition boundary at $\sigma_w^2 = 2$. Note that the condition for the phase transition turns out to be the same for ReLU networks with uncorrelated weights. However, for uncorrelated weights, the condition is invalid as $q_h^l(x)$ diverges for $\sigma_w^2 > 2$.



(a)



(b)

Figure 3.1: Phase diagram for ReLU networks with correlated Gaussian weights. (a) ReLU networks with positively correlated weights have two phases. It has, a 'bounded' phase where $q^*(x)$ is finite, and an 'unbounded' in which it diverges. The phase boundary shifts downwards relative to 2.1b. In both phases, any two signals will eventually become correlated (b) ReLU networks with anti-correlated weights have three phases. In addition to the phase boundary for the length (at some $\sigma_w^2 > 2$), a phase transition exists for the correlation coefficient. at $\sigma_w^2 = 2$.

In summary, the phase transition boundary for the length shifts depending on the correlation type, whereas the transition boundary for the correlation coefficient remains the same. The length's fixed point boundary shifts downwards for the positively correlated weights, resulting in similar phase diagram for uncorrelated weights (see Fig. 3.1a). Similar to ReLU networks with uncorrelated weights, we find that it does not have a chaotic phase in the unbounded phase. In contrast, an interesting situation occurs for the anti-correlated weights. The transition boundary for the length shifts upwards, creating an opportunity for a 'chaotic' phase (see Fig. 3.1b). Such a ReLU network has three phases. First, an 'ordered bounded' phase, where the length has a finite fixed point and correlations converge to one. Second an 'chaotic bounded' phase where the length still reaches a finite fixed point, but the correlations do not converge to one. Lastly, a 'chaotic unbounded' phase, where the length of the signal diverges. We demonstrate these results numerically in Fig. 3.2 for a correlation strength of $k = 100$. As predicted by the above equations, the stability of the fixed point $c_h^*(x_1, x_2) = 1$ changes at $\sigma_w^2 = 2$, and the length diverges for a higher value of σ_w^2. Therefore, the numerical experiments clearly validate the mean-field analysis.

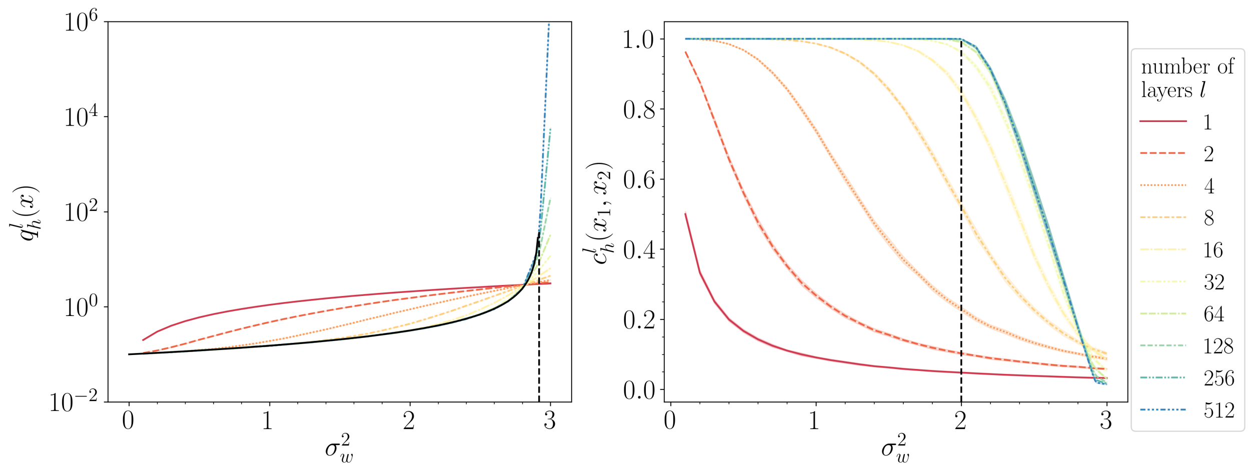


Figure 3.2: The above plots show the length and correlation coefficient of the signals after it has propagated through l layers in a ReLU network with anti-correlated weights. We calculate the length and the correlation coefficient for $M = 1024$ input signals, averaged over 40 networks with $N = 2048$ neurons in each layer. The dashed lines indicate the theoretical phase boundaries at $\sigma_w^2 = 2.92$ and $\sigma_w^2 = 2.0$ for the length and correlation coefficient. The solid black line in the first panel denotes the theoretical prediction for the length's fixed point. As in Fig. 2.2, the apparent crossing point before $\sigma_w^2 = 2.92$ is due to finite bias. As the critical boundaries do not depend on the variance of bias, we show results just for $\sigma_b^2 = 0.1$.

In conclusion, a ReLU network with anti-correlated weights can be more expressive by taking advantage of a chaotic phase, and it may be beneficial for a ReLU network to remain in this subspace. Thus, we propose initializing ReLU networks with anti-correlated weights at the phase transition boundary of the correlation coefficient $(\sigma_w^2, \sigma_b^2) = (2, 0)$. We call it Anti-Correlated Initialization (ACI). Section 3.2 shows numerical results for the training dynamics and performance of ReLU networks initialized with ACI for various tasks described in Section 2.3.

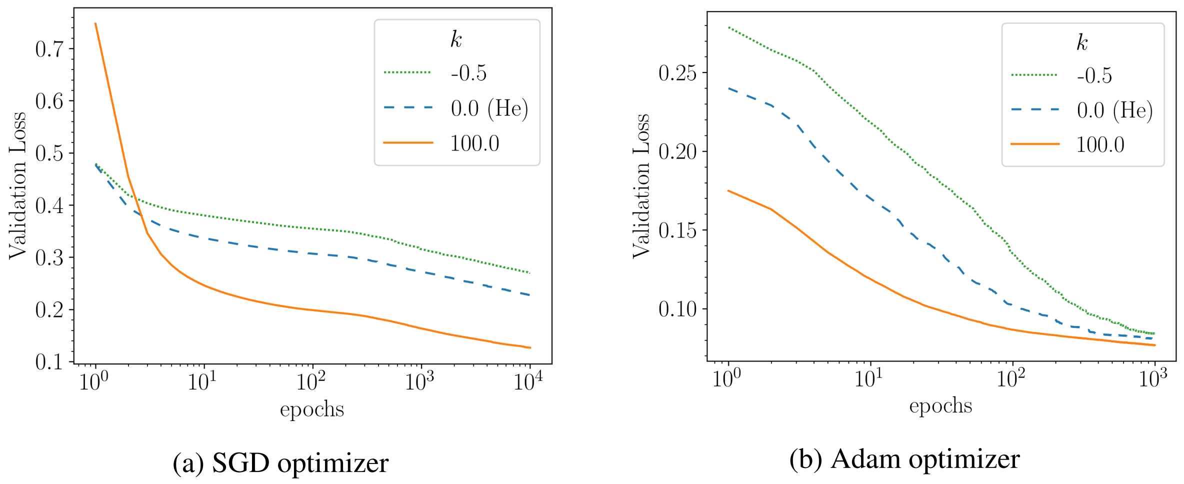
3.2 Training with Anti-Correlated Initialization

This section shows the training dynamics and performance of ReLU networks initialized with different weight correlation strength k on tasks described in Section 2.3. We choose three different correlation strengths; $k = 100$ induces anti-correlated weights, $k = -0.5$ produces positively correlated weights, and lastly, $k = 0$ corresponds to uncorrelated weights (He initialization). We train networks with two different optimization algorithms, SGD and Adam. For SGD, we train for 10^4

epochs, and for Adam, we train for 10^3 epochs.

3.2.1 Standard teacher task

Fig. 3.3a and 3.3b show the average validation loss for different correlation strengths trained using SGD and Adam algorithms. We observe that ReLU networks initialized with ACI ($k = 100$) train faster than He initialization ($k = 0$). In contrast, ReLU networks with positively correlated weights ($k = -0.5$) train even slower than He initialization.



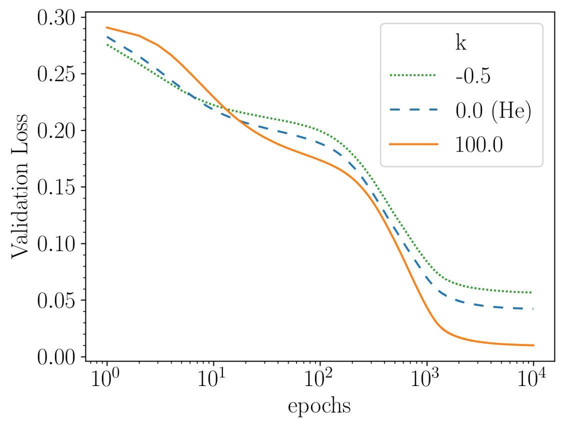
(a) SGD optimizer

(b) Adam optimizer

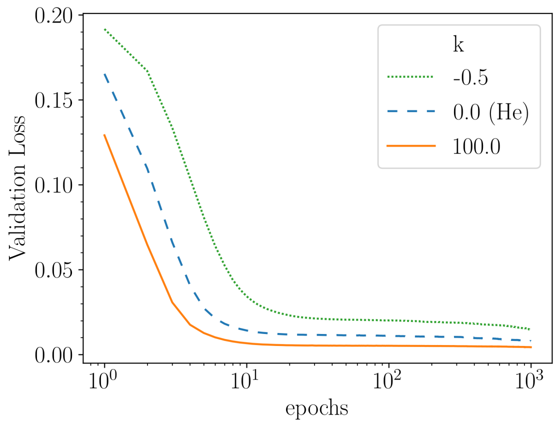
Figure 3.3: Average validation loss of 100 ReLU networks trained on the standard teacher task for different weight correlation strengths.

3.2.2 Simple teacher task

Fig. 3.4a and 3.4b show the average validation loss for the simple teacher task. We observe similar qualitative results as in the standard teacher task. For SGD, we observe an initial linear region in which all initialization schemes perform equally.



(a) SGD optimizer

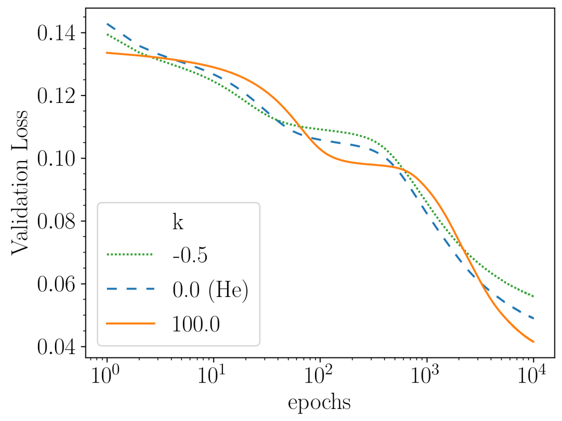


(b) Adam optimizer

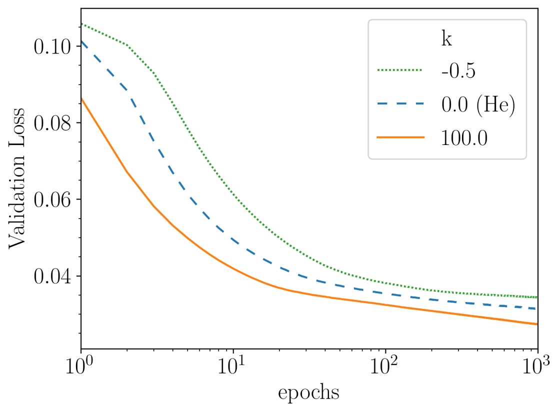
Figure 3.4: Average validation loss of 100 ReLU networks trained on the simple teacher task for different weight correlation strengths.

3.2.3 Complex teacher task

Fig. 3.5a and 3.5b show the average validation loss for a complex teacher task. For some intermediate regions, ACI performs worse than other initializations on training with SGD. The regions where ACI performs poorly shift depending on the complexity of the task.



(a) SGD optimizer



(b) Adam optimizer

Figure 3.5: Average validation loss of 100 ReLU networks trained on the complex teacher task for different weight correlation strengths.

In summary, ACI shows a definite advantage over He-initialization for most of the tasks. An exception is the complex task trained with SGD, where it shows comparable performance. Many alternatives are proposed to improve ReLU networks [24, 22, 25]. Of particular interest is Random Asymmetric Initialization (RAI), which replaces one of the weight/biases incoming to a neuron with a random variable drawn from a beta distribution to reduce the dead node probability. In the next section, begin by analyzing the correlation properties of RAI. We find that it only has an ordered phase as ReLU networks with uncorrelated weights. Therefore, there exists a possibility of inducing a chaotic phase using anti-correlated weights. Finally, we combine it with ACI to propose a new initialization scheme RAAI, which performs better than the best-known initialization schemes for ReLU networks.

3.3 Random Asymmetric Anti-correlated Initialization

This section explores a strategy of combining RAI (introduced by Ref. [22]) and ACI. We call this combined initialization Random Asymmetric Anti-correlated Initialization (RAAI). Before we begin, we analyze critical properties of RAI and show that similar to uncorrelated initialization RAI also has an ordered phase only. Therefore, there exists a possibility of obtaining a chaotic phase using anti-correlated weights. Finally, we describe RAAI and analyze its correlation properties. We find that similar to ACI, RAAI also has a chaotic phase which can be used to increase expressivity.

3.3.1 Random Asymmetric Initialization

We begin by analyzing critical properties of Random Asymmetric Initialization (RAI) proposed by Ref. [22] to reduce dead node probability. For ReLU networks with symmetric distributions (like Gaussian distributions with zero mean) for weights and biases, the probability of a dead node is half as ReLU outputs zero for negative inputs. RAI reduces the dead node probability by initializing one of the weights or bias incoming to the neuron by a distribution with positive support (like the beta distribution), resulting in a positive mean for the pre-activations. Note that any of the weights or the bias is randomly replaced. Thus, the weights and bias are treated on an equal footing. To simplify the notations, we incorporate the bias in the weight matrix by introducing a fictitious additional node with a constant value of 1, i.e.,

$$\mathbf{s}^l(x) = [1,\ \phi(\mathbf{h}^l(x))].$$

Ref. [22] propose to initialize RAI with a variance of $\sigma_w^2 = 0.36$ to ensure that the signal's length is bounded. However, the proposed variance does not represent the phase transition boundary for length, as in He initialization, and lies inside the bounded regime. This is because the fixed point stability condition is rather involved. In Appendix B.1, we estimate the transition boundary for the length under mean-field approximations. We found that the length has a finite non-zero fixed point if $\sigma_w^2 < 0.56$.

Similar to the analysis in Section 2.1, we analyze the correlation properties of RAI and found that $c_h^l(x_1, x_2) = 1$ is always a fixed point of the recursive map (see Appendix A.2). Finding the stability of the fixed point $c_h^*(x_1, x_2) = 1$, even with the mean-field assumptions is difficult, and we again obtain it approximately. In Appendix B.2, we estimate the transition boundary for the correlation coefficient under the mean-field approximations. Assuming $q^*(x)$ is finite, we show that ReLU networks with RAI do not have a chaotic phase, and any two signals propagating through a ReLU network become asymptotically correlated for all values of σ_w^2. In other words, $c^*(x1, x2) = 1$ is always a stable fixed point in the regime where signal strength is bounded. However, numerical results presented below indicate that the fixed point remains stable even in the unbounded phase. Note that we obtained similar results for ReLU networks initialized with uncorrelated Gaussian weights.

In Fig. 3.6, we show the numerical results for the fixed points of length and correlation coefficient. This suggests that the length remains finite for σ_w^2 up to 0.72 (first panel), and the $c_h^*(x_1, x_2) = 1$ is always a stable fixed point of the recursive map (second panel). Thus, the approximated results underestimate the length's critical point but correctly predict the stability of the correlation coefficient.

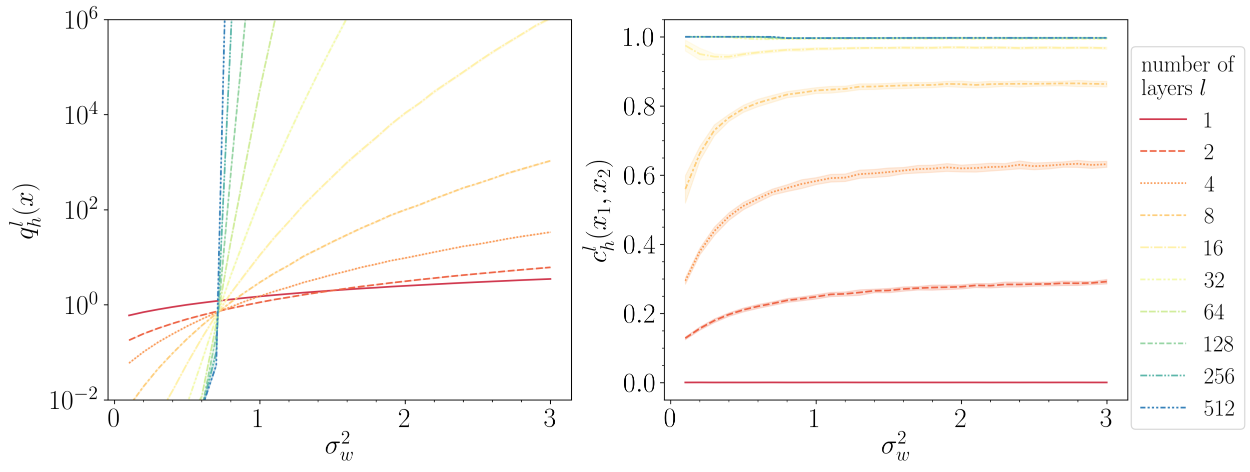


Figure 3.6: The above plots show the length and the correlation coefficient of the signals after propagating through l layers in a ReLU network initialized with RAI. We calculate the length and the correlation coefficient for $M = 1024$ input signals, averaged over 40 networks with $N = 2048$ neurons in each layer.

RAI focuses on decreasing the dead node probability to increase expressive power, whereas ACI uses anti-correlated weights to improve the expressivity. As RAI and ACI increase the expressivity of a ReLU network using different mechanisms, we explore the possibility of combining the two. We call it Random Anti-correlated Asymmetric Initialization (RAAI). In the following section, we describe RAAI and analyze its correlation properties.

3.3.2 Random Asymmetric Anti-correlated Initialization

To prepare weights drawn from RAAI, we take anti-correlated weights and biases incoming to a neuron (like Eqn. 3.1) and replace a randomly picked weight/bias with a random variable drawn from a beta distribution. Note that the weights incoming to different neurons are uncorrelated. Like ACI, we expect three phases, and we do observe the same in an approximate analysis of RAAI (see Appendix C). We find that the length diverges when $\sigma_w^2 > 1.75$, whereas the fixed point $c^*(x_1, x_2) = 1$ becomes unstable at $\sigma_w^2 = 1.41$. This again opens a possibility for a chaotic phase. We numerically verify these results in Fig. 3.7. We find that the length diverges for $\sigma_w^2 > 1.2$, whereas the correlation's fixed point is around $\sigma_w^2 = 0.9$. The numerical results qualitatively agree with our analysis, although the approximation in the analysis leads to an overestimation of the phase boundaries.

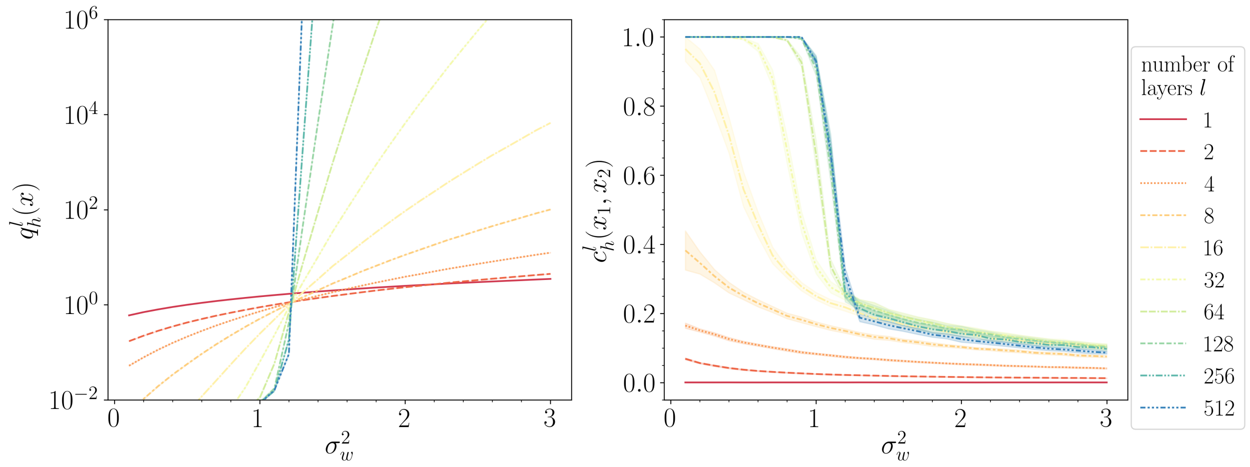


Figure 3.7: The above plots show length and the correlation coefficient of the signals after it has propagated through l layers in a ReLU network initialized with RAAI. We calculate the length and the correlation coefficient for $M = 1024$ input signals, averaged over 40 networks with $N = 2048$ neurons in each layer.

RAAI has a chaotic phase like ACI and a lower dead node probability like RAI. Thus, we expect RAAI to be a strong candidate for initializing ReLU networks. In Table 3.1, we summarize and compare different initialization schemes. For ACI, RAI, and RAAI, we scanned through the nearby parameter space to assure optimality of these parameters. In the next section, we analyze the training dynamics and performance of RAAI and compare it with other initialization schemes.

Initialization scheme	σ_w^2	σ_b^2	k	Chaotic phase	Dead node probability
He	2.0	0.0	0.0	No	0.5
ACI	2.0	0.0	100.0	Yes	0.5
RAI	0.36	0.36	0.0	No	0.36
RAAI	0.90	0.90	100.0	Yes	0.36

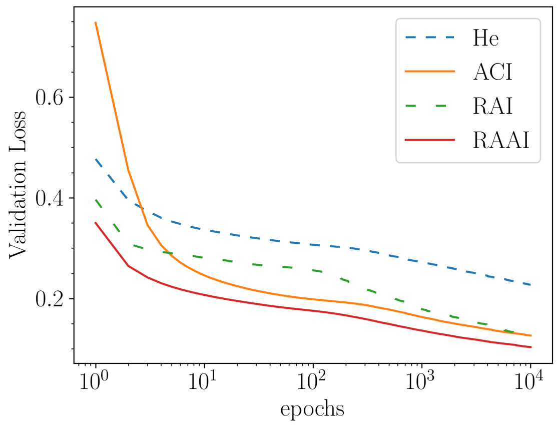
Table 3.1: Comparison between different initialization schemes for ReLU networks. RAI and RAAI both have a lower dead node probability than the other two symmetric initializations. ACI and RAAI have a chaotic phase, whereas the other two do not. Thus, RAAI has both a chaotic phase and a lower dead node probability. The dead node probabilities are calculated numerically for input signals drawn from the standard normal distribution.

3.4 Training with Random Asymmetric Anti-correlated Initialization

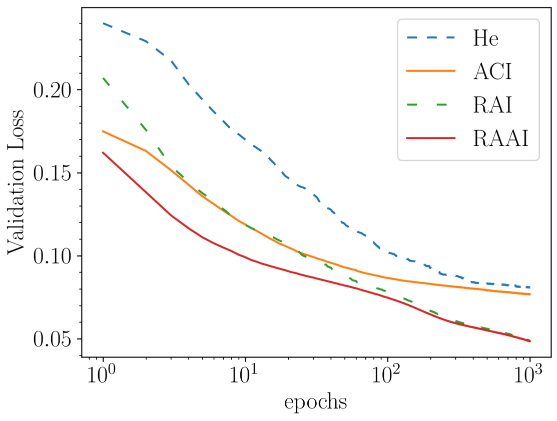
This section compares the performance of RAAI with other initialization schemes listed in Table 3.1 on tasks described in Section 2.3.

3.4.1 Standard teacher task

Fig. 3.8a and 3.8b show the average validation loss for different initialization schemes described in Table 3.1 trained using SGD and Adam algorithms. We observe that RAAI performs on par with or better than all other initialization schemes, whereas the relative performance of RAI and ACI depends on the optimization algorithm.



(a) SGD optimizer

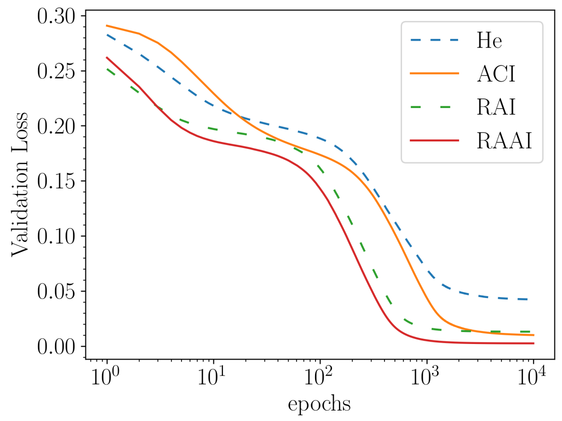


(b) Adam optimizer

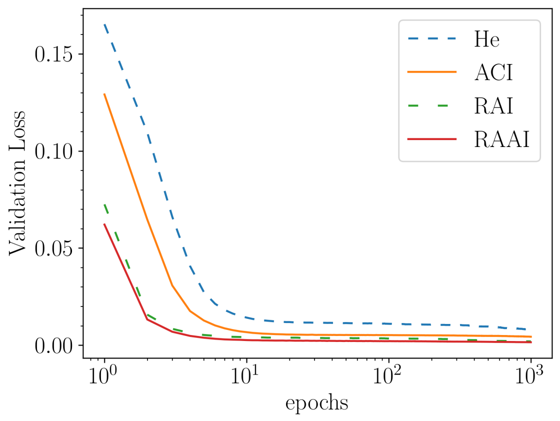
Figure 3.8: Average validation loss for ReLU networks trained on the standard teacher task for different initialization schemes.

3.4.2 Simple teacher task

Fig. 3.9a and 3.9b show the average validation loss for the simple teacher task for different initialization schemes. Similar to the standard teacher task, RAAI performs better than or on par with other initialization schemes. In between RAI and ACI, the former performs better on using either optimizer.



(a) SGD optimizer

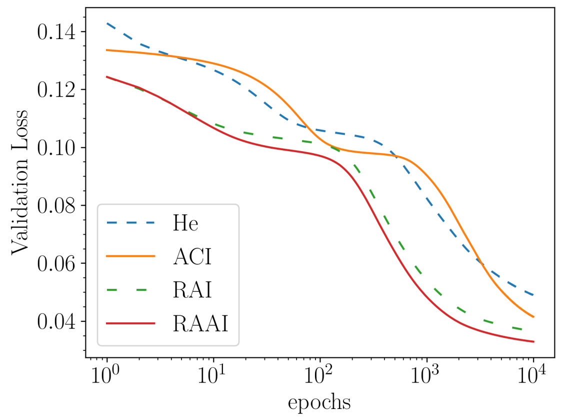


(b) Adam optimizer

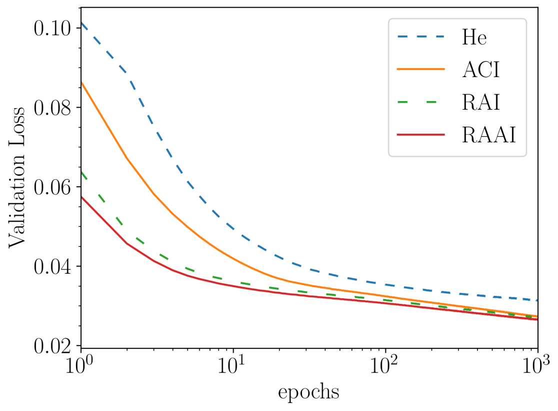
Figure 3.9: Average validation loss for ReLU networks trained on the simple teacher task for different initialization schemes.

3.4.3 Complex teacher task

Fig. 3.10a and 3.10b show the average validation loss for a complex teacher task. Again, RAAI performs better than or on par with all other initialization schemes.



(a) SGD optimizer



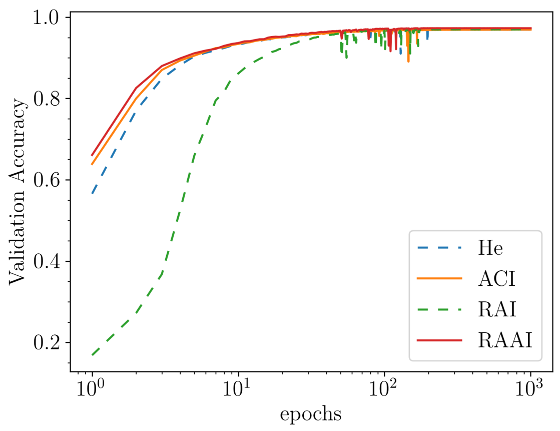
(b) Adam optimizer

Figure 3.10: Average validation loss for ReLU networks trained on the complex teacher task for different initialization schemes.

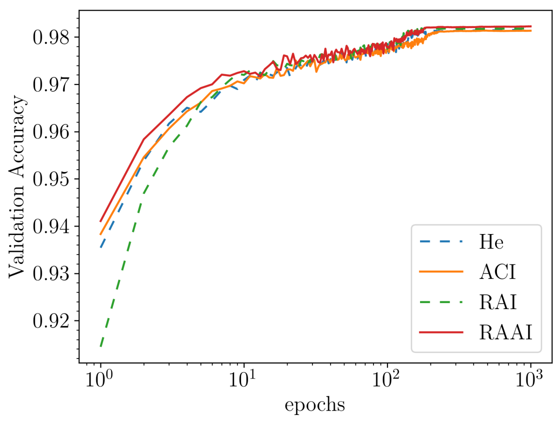
It may be of concern that initialization in an expressive subspace of weights might lead to overfitting. We trained a neural network with $L = 20$ layers and found that ACI starts to overfit, but this can be avoided by reducing the value of the correlation strength k. This leads to another parameter to be tuned. On the other hand, RAAI does not show any overfitting signs and performs consistently better than other initialization schemes on training deeper networks. We believe that overfitting in ACI is related to correlations vanishing to zero at high variance (see Fig. 3.2). In contrast, for RAAI, correlations do not vanish to zero at high variance (see Fig. 3.7).

3.4.4 MNIST task

Fig. 3.11a and 3.11b show the average validation loss for the MNIST task. Again, RAAI performs better than all other initialization schemes, whereas, RAI trains slower even than He initialization.



(a) SGD optimizer

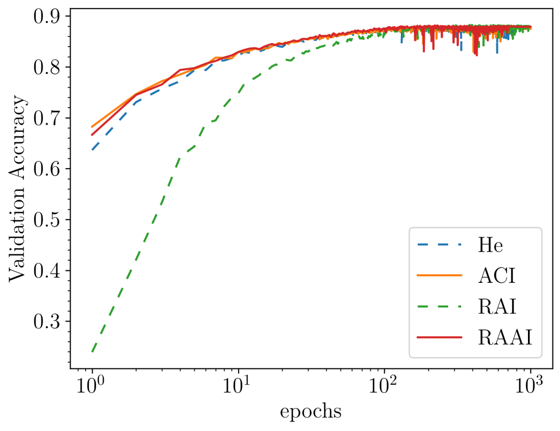


(b) Adam optimizer

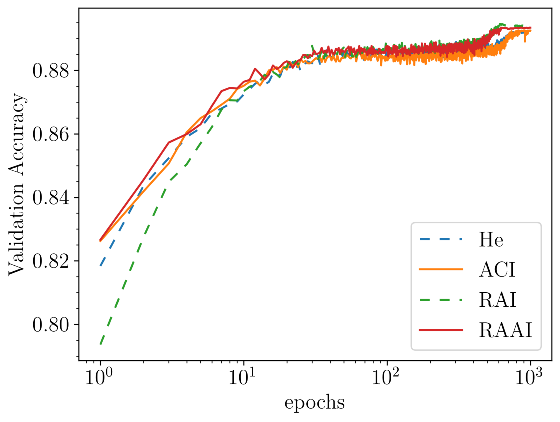
Figure 3.11: Average validation accuracy of 10 ReLU networks trained on the MNIST task for different initialization schemes.

3.4.5 Fashion-MNIST task

Fig. 3.12a and 3.12b show the average validation loss for the Fashion-MNIST task. RAAI performs better than all other initialization schemes, whereas, RAI trains slower even than He initialization.



(a) SGD optimizer



(b) Adam optimizer

Figure 3.12: Average validation accuracy of 10 ReLU networks trained on the Fashion-MNIST task for different initialization schemes.

Chapter 4

Discussion and conclusion

This chapter begins by discussing various questions raised in Sections 1.1 and 1.2 regarding the expressivity of deep neural networks. Next, we discuss the implications and limitations of the new results presented in this thesis. Lastly, we provide an outlook on a few open problems of deep neural networks.

Implications of the mean-field analysis on the expressivity of deep neural networks

This section discusses the implication of the mean-field analysis on the expressivity of deep neural networks. In particular, we discuss if a deeper network always helps solve a more complex problem and why the phase transition boundary corresponds to the optimal information flow.

Is a deeper network always better? This question is often raised, and practitioners always try to train deeper networks to solve complex problems. However, the mean-field theory of signal propagation suggests that the expressivity of a deep network increases/decreases as depth depending on the chaotic/ordered phase. In an ordered phase, two signals become asymptotically correlated; thus, the expressivity decreases as a function of depth. In contrast, two signals become increasingly different if the correlation coefficient's fixed point is zero; therefore, the expressivity increases as depth. Therefore, increasing the depth of a network does not necessarily help in solving complex problems.

The mean-field analysis allows us to find optimal conditions for information flow through the network. Both the ordered and the chaotic phase modify the correlations between the input signals. If this modification is too drastic, it might lead to training failure. For example, consider a cat vs. dog image classification task. A network may find it difficult to differentiate between a cat and a dog image in an ordered phase, whereas it may face difficulties in keeping two cat images in the same class in a chaotic phase. However, the depth scale $\xi_c^{-1} = -\log \chi_1$ corresponding to the asymptotic expansion, $|c_h^l(x_1,x_2) - 1| \sim e^{-l/\xi_c}$, diverges at the phase transition boundary as $\chi_1 \to 1$, and the input correlations sustain for an extended depth. Thus, the phase transition boundary parameters correspond to optimal information flow through the network. Note that these problems occur only in deep neural networks.

Implications and limitations of the new results

In this work, we analyzed the evolution of correlation between signals propagating through a ReLU network with correlated weights using the mean-field theory of signal propagation. Multiple studies show that ReLU networks with uncorrelated weights are biased towards computing simpler functions, but ReLU networks do perform complex tasks in practice. Unlike ReLU networks with uncorrelated weights, ReLU networks with anti-correlated weights reaching a node have a chaotic phase where correlation saturates below unity. This suggests that such networks can exhibit higher expressivity. Although we have focused on the ReLU networks in this study, anti-correlation in weights may be helpful in general. Networks with other non-linear activation functions like tanh, SELU, and sigmoid have a chaotic phase even with uncorrelated weights. Nevertheless, the weight correlations in them may still help to tune the phase boundaries and expressivity of the networks.

We further investigated the possibility that ReLU networks with the enhanced expressivity may prove beneficial in faster learning. Comparison of training and test performance of networks in a range of teacher-student setups clearly showed that networks with anti-correlated weights learn faster. While ACI shows better learning performance in general, it shows poor performance with SGD during an intermediate learning stage when the teacher network has a relatively higher expressivity. We believe that this may be due to the system getting stuck in local minima. This is consistent with the absence of a similar regime on training with Adam optimizer. On training deeper networks with ACI, we found that it overfits, but this can be avoided by fine-tuning correlation strength k.

As anti-correlated weights exhibit faster training for ReLU networks, we explored a strategy of favoring anti-correlation in the training procedure. We investigated improvement in training time from a regularization term in the loss function that favors anti-correlated weights. However, our attempts did not show any systematic results.

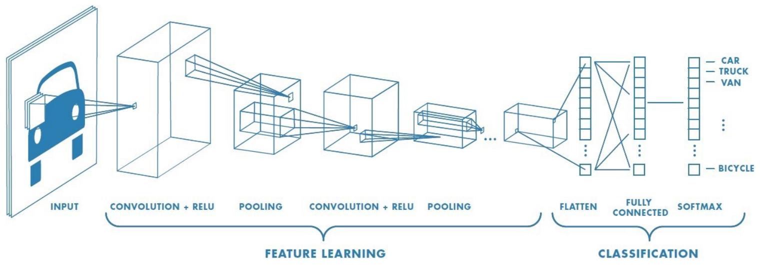
We compared ACI with a recently proposed initialization scheme called RAI, which introduces a systematic asymmetry (around the mean) in the weights to decrease node probability. We find that the relative performance between RAI and ACI depends on the task and the optimization algorithm. RAI improves expressivity by reducing the dead node probability, whereas ACI achieves the same by inducing a chaotic phase. As RAI and ACI rely on different mechanisms, we explored a strategy of combining the two initialization schemes. We analyzed the correlation properties of RAAI and found that it has a chaotic phase like ACI. We demonstrated that RAAI leads to faster training and learning than the best-known methods on various teacher tasks and image classification tasks. For different initialization schemes, the behavior of the training dynamics at large epochs may depend on the optimizer and training data. However, RAAI shows a definite advantage over other schemes when using the SGD optimizer, especially in early training epochs. In addition to faster training, RAAI also shows no sign of overfitting and improves on the simpler strategy that relies only on anti-correlations.

Further work and outlook

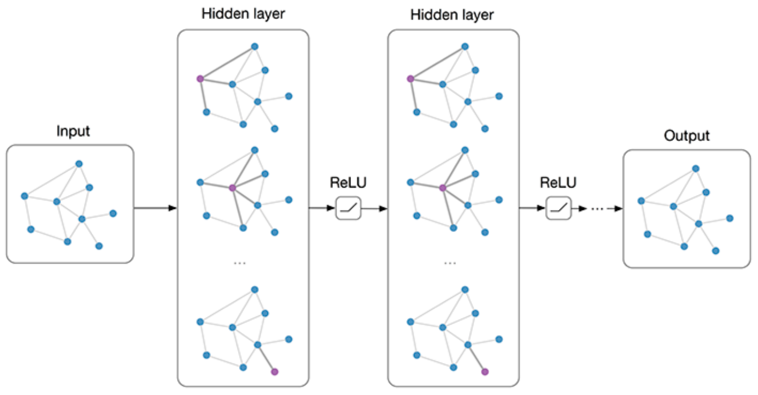
In this thesis, we focused on the evolution of correlations between signals propagating through infinitely wide neural networks. In practice, neural networks can only be finite. Therefore, one following problem is to study the effect of finite width and depth. Ref. [44] studied large deviations to weight perturbations applied to deep neural networks. However, there are no studies on the phase diagram of signal propagation in deep neural networks with finite width and depth.

We also completely ignored analyzing the training dynamics in this study. Ref. [57] observed that the weights of a wide network do not change significantly after training. In this lazy regime, the evolution of the output function of a wide network can be approximated by Taylor series analysis. This approximation is called Neural Tangent Kernel. Under this limit, a neural network can be approximated as an evolving Gaussian kernel, and the continuous version of gradient descent finds parameters corresponding to zero training loss. We would consider extending this analysis for ReLU networks with correlated weights to find theoretical evidence for faster training with

ACI. Other open problems regarding the training dynamics include the theoretical analysis of the learning rate, the mini-batch procedure, regularization procedures like dropout, and atypical loss functions and optimizers, to name a few.



(a) A convolutional neural network [a]



(b) A graph neural network [b]

[a]https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53

[b]https://tkipf.github.io/graph-convolutional-networks/

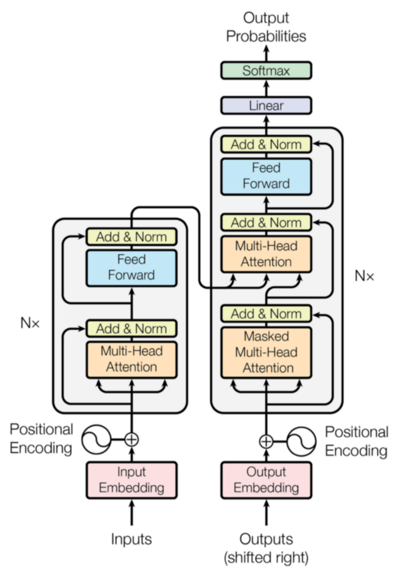


Figure 1: The Transformer - model architecture.

(c) A transformer [a]

[a]https://medium.com/inside-machine-learning/what-is-a-transformer-d07dd1fbec04

Figure 4.1: Some recently introduced architectures.

In the last few years, several new architectures and new learning rules have been proposed, which have replaced feed-forward networks in the supervised learning setting. Architectures like Convolutional Neural Networks [11], Graph Neural Networks [58], and Transformers [59] (see Figure 4.1) provide a significant advantage over feed-forward networks. However, these models have complex architectures and are challenging to interpret. The mean-field formalism can still be applied to convolutional neural networks and graph neural networks under simplified settings (for example, see Ref. [60]). In contrast, the theoretical analysis of the expressivity of recent architectures like Transformers is still an open question. Moreover, feed-forward neural networks are part of many of these complex architectures. Therefore, it is still relevant to study feed-forward neural networks to have a conceptual understanding of these complex architectures.

The machine learning community is presently moving away from the supervised learning setup because the training loss observed in supervised learning provides no information about the generalization error. Neural networks can achieve zero training loss even on random labels when the testing loss can still be considerably large. New learning rules under the name of self-supervised learning are introduced, which can avoid this fate. However, a one-for-all learning rule is still not developed. Nevertheless, the new learning rules still use neural networks, and questions about the expressivity of neural networks remain relevant, and we hope that this study will continue to be applicable in the future.

Bibliography

[1] Xavier Glorot, Antoine Bordes, and Yoshua Bengio. Deep sparse rectifier neural networks. In Geoffrey Gordon, David Dunson, and Miroslav Dudík, editors, *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, volume 15 of *Proceedings of Machine Learning Research*, pages 315–323, Fort Lauderdale, FL, USA, 11–13 Apr 2011. JMLR Workshop and Conference Proceedings.

[2] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 25, pages 1097–1105. Curran Associates, Inc., 2012.

[3] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. *IEEE International Conference on Computer Vision (ICCV 2015)*, 1502, 02 2015.

[4] Andrew L. Maas, Awni Y. Hannun, and Andrew Y. Ng. Rectifier nonlinearities improve neural network acoustic models. In *in ICML Workshop on Deep Learning for Audio, Speech and Language Processing*, 2013.

[5] L. Tóth. Phone recognition with deep sparse rectifier neural networks. *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 6985–6989, 2013.

[6] G. Hinton, L. Deng, D. Yu, G. E. Dahl, A. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. N. Sainath, and B. Kingsbury. Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *IEEE Signal Processing Magazine*, 29(6):82–97, 2012.

[7] David Silver, Aja Huang, Christopher Maddison, Arthur Guez, Laurent Sifre, George Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, Sander Dieleman, Dominik Grewe, John Nham, Nal Kalchbrenner, Ilya Sutskever, Timothy Lillicrap, Madeleine Leach, Koray Kavukcuoglu, Thore Graepel, and Demis Hassabis. Mastering the game of go with deep neural networks and tree search. *Nature*, 529:484–489, 01 2016.

[8] T Ciodaro, Mr D Dd, Joao Seixas, and Denis Oliveira Damazio. Online particle detection with neural networks based on topological calorimetry information. *Journal of Physics: Conference Series*, 368, 06 2012.

[9] J. Ma, R. Sheridan, Andy Liaw, George E. Dahl, and V. Svetnik. Deep neural nets as a method for quantitative structure-activity relationships. *Journal of chemical information and modeling*, 55 2:263–74, 2015.

[10] Andrew Senior, Richard Evans, John Jumper, James Kirkpatrick, Laurent Sifre, Tim Green, Chongli Qin, Augustin Žídek, Alexander Nelson, Alex Bridgland, Hugo Penedones, Stig Petersen, Steve Crossan, Pushmeet Kohli, David Jones, David Silver, Koray Kavukcuoglu, and Demis Hassabis. Improved protein structure prediction using potentials from deep learning. *Nature*, 577:1–5, 01 2020.

[11] K. O'Shea and Ryan Nash. An introduction to convolutional neural networks. *ArXiv*, abs/1511.08458, 2015.

[12] Yong Yu, Xiaosheng Si, Changhua Hu, and Jianxun Zhang. A Review of Recurrent Neural Networks: LSTM Cells and Network Architectures. *Neural Computation*, 31(7):1235–1270, 07 2019.

[13] Yann LeCun, Y. Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521:436–44, 05 2015.

[14] Yasaman Bahri, Jonathan Kadmon, Jeffrey Pennington, Sam S. Schoenholz, Jascha Sohl-Dickstein, and Surya Ganguli. Statistical mechanics of deep learning. *Annual Review of Condensed Matter Physics*, 11(1):501–528, 2020.

[15] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2015.

[16] RMSprop. `https://www.cs.toronto.edu/~tijmen/csc321/slides/lecture_slides_lec6.pdf`. Accessed: 2021-04-30.

[17] Matthew D. Zeiler. Adadelta: An adaptive learning rate method. *ArXiv*, abs/1212.5701, 2012.

[18] Sebastian Ruder. An overview of gradient descent optimization algorithms. 09 2016.

[19] Nitish Keskar, Dheevatsa Mudigere, Jorge Nocedal, Mikhail Smelyanskiy, and Ping Tang. On large-batch training for deep learning: Generalization gap and sharp minima. 09 2016.

[20] Simon Du, Jason Lee, Haochuan Li, Liwei Wang, and Xiyu Zhai. Gradient descent finds global minima of deep neural networks. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 1675–1685. PMLR, 09–15 Jun 2019.

[21] Rupesh Srivastava, Klaus Greff, and Jürgen Schmidhuber. Training very deep networks. *2015 Neural Information Processing Systems (NIPS 2015 Spotlight)*, 07 2015.

[22] Lu Lu, Yeonjong Shin, Yanhui Su, and George Karniadakis. Dying relu and initialization: Theory and numerical examples. *Communications in Computational Physics*, 28:1671–1706, 11 2020.

[23] Prajit Ramachandran, Barret Zoph, and Quoc V. Le. Searching for activation functions, 2018.

[24] Ludovic Trottier, P. Giguère, and B. Chaib-draa. Parametric exponential linear unit for deep convolutional neural networks. *2017 16th IEEE International Conference on Machine Learning and Applications (ICMLA)*, pages 207–214, 2017.

[25] Djork-Arné Clevert, Thomas Unterthiner, and Sepp Hochreiter. Fast and accurate deep network learning by exponential linear units (elus). In Yoshua Bengio and Yann LeCun, editors, *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*, 2016.

[26] Günter Klambauer, Thomas Unterthiner, Andreas Mayr, and Sepp Hochreiter. Self-normalizing neural networks. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30, pages 971–980. Curran Associates, Inc., 2017.

[27] Dan Hendrycks and Kevin Gimpel. Gaussian error linear units (gelus). *arXiv: Learning*, 2016.

[28] Kaiming He, X. Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016.

[29] Lei Jimmy Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. Layer normalization. *CoRR*, abs/1607.06450, 2016.

[30] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In Francis Bach and David Blei, editors, *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pages 448–456, Lille, France, 07–09 Jul 2015. PMLR.

[31] Tim Salimans and Diederik P. Kingma. Weight normalization: A simple reparameterization to accelerate training of deep neural networks. In *NIPS*, 2016.

[32] D. Ulyanov, A. Vedaldi, and V. Lempitsky. Instance normalization: The missing ingredient for fast stylization. *ArXiv*, abs/1607.08022, 2016.

[33] Yuxin Wu and Kaiming He. Group normalization. In *ECCV*, 2018.

[34] Dmytro Mishkin and Jiri Matas. All you need is a good init. *CoRR*, abs/1511.06422, 2016.

[35] Yurii Nesterov. *Introductory Lectures on Convex Optimization: A Basic Course*. Springer Publishing Company, Incorporated, 1 edition, 2014.

[36] Xavier Glorot, Antoine Bordes, and Yoshua Bengio. Deep sparse rectifier neural networks. In Geoffrey Gordon, David Dunson, and Miroslav Dudík, editors, *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, volume 15 of *Proceedings of Machine Learning Research*, pages 315–323, Fort Lauderdale, FL, USA, 11–13 Apr 2011. JMLR Workshop and Conference Proceedings.

[37] Andrew M. Saxe, James L. McClelland, and Surya Ganguli. Exact solutions to the nonlinear dynamics of learning in deep linear neural networks. In Yoshua Bengio and Yann LeCun, editors, *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*, 2014.

[38] Ben Poole, Subhaneil Lahiri, Maithra Raghu, Jascha Sohl-Dickstein, and Surya Ganguli. Exponential expressivity in deep neural networks through transient chaos. In D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 29, pages 3360–3368. Curran Associates, Inc., 2016.

[39] Maithra Raghu, Ben Poole, Jon Kleinberg, Surya Ganguli, and Jascha Sohl Dickstein. On the expressive power of deep neural networks. In *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, ICML'17, page 2847–2854. JMLR.org, 2017.

[40] Samuel S. Schoenholz, Justin Gilmer, Surya Ganguli, and Jascha Sohl-Dickstein. Deep information propagation. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017.

[41] Jaehoon Lee, Jascha Sohl-dickstein, Jeffrey Pennington, Roman Novak, Sam Schoenholz, and Yasaman Bahri. Deep neural networks as gaussian processes. In *International Conference on Learning Representations*, 2018.

[42] Soufiane Hayou, Arnaud Doucet, and Judith Rousseau. On the selection of initialization and activation function for deep neural networks, 2019.

[43] Bo Li and David Saad. Exploring the function space of deep-learning machines. *Phys. Rev. Lett.*, 120:248301, Jun 2018.

[44] Bo Li and David Saad. Large deviation analysis of function sensitivity in random deep neural networks. *Journal of Physics A: Mathematical and Theoretical*, 53(10):104002, feb 2020.

[45] Hartmut Maennel, O. Bousquet, and S. Gelly. Gradient descent quantizes relu network features. *ArXiv*, abs/1803.08367, 2018.

[46] Boris Hanin. Which neural net architectures give rise to exploding and vanishing gradients? In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 31, pages 582–591. Curran Associates, Inc., 2018.

[47] Giacomo De Palma, Bobak Kiani, and Seth Lloyd. Random deep neural networks are biased towards simple functions. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32, pages 1964–1976. Curran Associates, Inc., 2019.

[48] Nasim Rahaman, Aristide Baratin, Devansh Arpit, Felix Draxler, Min Lin, Fred Hamprecht, Yoshua Bengio, and Aaron Courville. On the spectral bias of neural networks, 2019.

[49] Juncai He, Lin Li, Jinchao Xu, and Chunyue Zheng. Relu deep neural networks and linear finite elements. *Journal of Computational Mathematics*, 38(3):502–527, 2020.

[50] Guillermo Valle-Perez, Chico Q. Camargo, and Ard A. Louis. Deep learning generalizes because the parameter-function map is biased towards simple functions. In *International Conference on Learning Representations*, 2019.

[51] B. Hanin and D. Rolnick. Deep relu networks have surprisingly few activation patterns. In *NeurIPS*, 2019.

[52] Wenling Shang, Kihyuk Sohn, Diogo Almeida, and Honglak Lee. Understanding and improving convolutional neural networks via concatenated rectified linear units. In *Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48*, ICML'16, page 2217–2225. JMLR.org, 2016.

[53] Soufiane Hayou, Arnaud Doucet, and Judith Rousseau. On the impact of the activation function on deep neural networks training. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 2672–2680. PMLR, 09–15 Jun 2019.

[54] H. S. Seung, H. Sompolinsky, and N. Tishby. Statistical mechanics of learning from examples. *Phys. Rev. A*, 45:6056–6091, Apr 1992.

[55] Sebastian Goldt, Marc Mezard, Florent Krzakala, and Lenka Zdeborova. Modeling the influence of data structure on learning in neural networks: The hidden manifold model. *Physical Review X*, 10, 12 2020.

[56] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. Tensorflow: Large-scale machine learning on heterogeneous distributed systems, 2015.

[57] Arthur Jacot, Franck Gabriel, and Clement Hongler. Neural tangent kernel: Convergence and generalization in neural networks. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018.

[58] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and Philip Yu. A comprehensive survey on graph neural networks, 01 2019.

[59] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

[60] Lechao Xiao, Yasaman Bahri, Jascha Sohl-Dickstein, Samuel Schoenholz, and Jeffrey Pennington. Dynamical isometry and a mean field theory of CNNs: How to train 10,000-layer vanilla convolutional neural networks. In Jennifer Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 5393–5402. PMLR, 10–15 Jul 2018.

[61] Vinod Nair and Geoffrey E. Hinton. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th International Conference on International Conference on Machine Learning*, ICML'10, page 807–814, Madison, WI, USA, 2010. Omnipress.

[62] C. Szegedy, Wei Liu, Yangqing Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1–9, 2015.

[63] Abien Fred Agarap. Deep learning using rectified linear units (relu). *ArXiv*, abs/1803.08375, 2018.

[64] John C. Duchi, Elad Hazan, and Y. Singer. Adaptive subgradient methods for online learning and stochastic optimization. In *J. Mach. Learn. Res.*, 2011.

[65] K. Fukushima. Visual feature extraction by a multilayered network of analog threshold elements. *IEEE Transactions on Systems Science and Cybernetics*, 5(4):322–333, 1969.

[66] Kunihiko Fukushima and Sei Miyake. Neocognitron: A self-organizing neural network model for a mechanism of visual pattern recognition. In Shun-ichi Amari and Michael A. Arbib, editors, *Competition and Cooperation in Neural Nets*, pages 267–285, Berlin, Heidelberg, 1982. Springer Berlin Heidelberg.

[67] Alireza Seif, M. Hafezi, and C. Jarzynski. Machine learning the thermodynamic arrow of time. *Nature Physics*, 17:105–113, 2019.

Appendix A

Derivation of the length and covariance map for RAI

To obtain weights drawn from RAI distribution, we initialize the weights and bias incoming to a node with a Gaussian distribution $\sim \mathcal{N}(0, \sigma_w^2)$ and replace one weight or the bias with a beta distribution random variable. The weights and bias are treated on an equal footing. Therefore, to simplify the notations, we incorporate the bias in the weight matrix by introducing a fictitious additional node with a constant value of one, i.e.,

$$\mathbf{s}^l(x) = [1,\ \phi(\mathbf{h}^l(x))].$$

Now, the node index starts from 0 instead of 1. Furthermore, we assume that $N_l = N$ at each layer. The evolution equation is now given by

$$\mathbf{h}^l(x) = \mathbf{W}^l \cdot \mathbf{s^{l-1}}(x).$$

It is easier to track the evolution of correlations using the activations instead of the pre-activations for RAI-type initializations for which the analytical form of the distribution is unknown. So, we define a couple of covariance matrices over activations which will come in handy

$$
\begin{aligned}
q_s^l(x_1,x_2) &= \frac{1}{N+1}\sum_{i=0}^{N} s_i^l(x_1)s_i^l(x_2) \\
q_{-k_j^l}^l(x_1,x_2) &= \frac{1}{N}\sum_{t\neq k_j^l}^{N} s_t^l(x_1)s_t^l(x_2)\ ,
\end{aligned}
$$

where k_j^l tags variables associated with the special weight. We will use the notations $q_s^l(x) = q_s^l(x,x)$ and $q_{-k}^l(x) = q_{-k}^l(x,x)$. The corresponding correlation coefficients are given by,

$$
\begin{aligned}
c_s^l(x_1,x_2) &= \frac{q_s^l(x_1,x_2)}{\sqrt{q_s^l(x_1)\, q_s^l(x_2)}} \\
c_{-k_j^l}^l(x_1,x_2) &= \frac{q_{-k_j^l}^l(x_1,x_2)}{\sqrt{q_{-k_j^l}^l(x_1)\, q_{-k_t^l}^l(x_2)}}
\end{aligned}
$$

A.1 Derivation of the length map for RAI

Given $\boldsymbol{h}^{l-1}(x)$, we can view $h_j^l(x)$ as

$$
h_j^l(x) = \sigma_w \sqrt{q_{-k_j^{l-1}}^{l-1}(x)}\, z + s_{k_j^{l-1}}^{l-1}(x)\, u,
$$

where $z \sim \mathcal{N}(0,1)$ and $u \sim \beta(2,1)$. By applying the activation function and squaring it, we obtain

$$
\phi(h_j^l(x))^2 = \phi\left(\sigma_w \sqrt{q_{-k_j^{l-1}}^{l-1}(x)}\, z + s_{k_j^{l-1}}^{l-1}(x)\, u\right)^2 .
$$

Next, we take an average over the weights and the special weight to get

$$\left\langle \phi(h_j^l(x))^2 | \boldsymbol{h}^{l-1}(x) \right\rangle = \sum_{k_j^{l-1}=0}^{N} \frac{1}{N+1} \int dz\, du\, f(z)\, f(u)\, \phi\left(\sigma_w \sqrt{q^{l-1}_{-k_j^{l-1}}(x)}\, z + s^{l-1}_{k_j^{l-1}}(x)\, u\right)^2 ,$$

where $f(u) \sim \beta(2,1)$, and $f(z) \sim \mathcal{N}(0,1)$. Finally, we sum over all nodes and re-write the equation in terms of the overlap

$$\left\langle q_s^l(x) | \boldsymbol{h}^{l-1}(x) \right\rangle = \frac{1}{N+1}\left[1 + \sum_{j=1}^{N} \sum_{k_j^{l-1}=0}^{N} \frac{1}{N+1} \int dz\, du\, f(z) f(u)\, \phi\left(\sigma_w \sqrt{q^{l-1}_{-k_j^{l-1}}(x)}\, z + s^{l-1}_{k_j^{l-1}}(x)\, u\right)^2\right]. \tag{A.1}$$

A.2 Derviation of the covariance map for RAI

The covariance map can be derived similarly, with a key difference of correlations between the pre-activations incoming to the node. For two input signals x_1 and x_2, the covariance map reads

$$\begin{aligned}\left\langle q_s^l(x_1,x_2) | \boldsymbol{h}^{l-1}(x_1), \boldsymbol{h}^{l-1}(x_2) \right\rangle = \frac{1}{N+1}\Bigg[1 + \sum_{j=1}^{N} \sum_{k_j^{l-1}=0}^{N} \frac{1}{N+1} \int dy_1 dy_2 du\, f(y_1,y_2)\, f(u)\, \times \\ \times\ \phi\left(\sigma_w y_1 + s^{l-1}_{k_j^{l-1}}(x_1)\, u\right) \phi\left(\sigma_w y_2 + s^{l-1}_{k_j^{l-1}}(x_2)\, u\right)\Bigg],\end{aligned} \tag{A.2}$$

where $f(y_1,y_2)$ is the joint Gaussian distribution of y_1 and y_2, with a covariance matrix given by

$$\Sigma^{l-1}_{k_j^l}(x_1,x_2) = \begin{bmatrix} q^{l-1}_{-k_j^{l-1}}(x_1) & q^{l-1}_{-k_j^{l-1}}(x_1,x_2) \\ q^{l-1}_{-k_j^{l-1}}(x_1,x_2) & q^{l-1}_{-k_j^{l-1}}(x_2) \end{bmatrix}.$$

We can re-write Eqn. A.2 in terms of $c^{l}_{-k^{l}_{j}}(x_1,x_2)$

$$\left\langle q^{l}_{s}(x_1,x_2)|\boldsymbol{h}^{l-1}(x_1),\boldsymbol{h}^{l-1}(x_2)\right\rangle = \frac{1}{N+1}\left[1+\sum_{j}\sum_{k^{l-1}_{j}}\frac{1}{N+1}\int dz_1\, dz_2\, du\, f(z_1)f(z_2)\, f(u)\, \phi\,(v_1)\,\phi\,(v_2)\right]$$

$$v_1 = \sigma_w\sqrt{q^{l-1}_{-k^{l-1}_{j}}(x_1)}\, z_1 + s^{l-1}_{k^{l-1}_{j}}(x_1)\, u$$

$$v_2 = \sigma_w\sqrt{q^{l-1}_{-k^{l-1}_{j}}(x_2)}\left[c^{l}_{-k^{l}_{j}}(x_1,x_2)z_1 + \sqrt{1-(c^{l}_{-k^{l}_{j}}(x_1,x_2))^2}\, z_2\right] + s^{l-1}_{k^{l-1}_{j}}(x_2)\, u$$

where $f(z_1) \sim f(z_2) \sim \mathcal{N}(0,1)$ are standard Gaussian distributions. As suggested by Ref. [38], we can find the fixed point of the correlation map under the assumption that the length $q^{l}_{h}(x)$ has reached its fixed point. Under this assumption, it is easy to see that $c^{l}_{h}(x_1,x_2) = 1$ is a fixed point of the correlation map.

Appendix B

Stability of the fixed points for the length and correlation maps for RAI

B.1 Stability of the fixed point for the length map for RAI

The derivation of the analytical form of the length map (Eqn. A.1) is involved, and only bounds to the map have been derived [22]. Inspired by the analytical form of the length map for the anti-correlated initialization and the analysis done by Ref. [22], we assume that the length map has a linear dependence on $q_s^{l-1}(x)$. Under this assumption, we can find the stability of the fixed point of the length map by taking a derivative with respect to $q_s^{l-1}(x)$. However, another problem exists. While taking a derivative, we have to encounter derivatives of the form

$$\frac{\partial s^{l-1}_{-k_j^{l-1}}(x)}{\partial q_s^{l-1}(x)}.$$

To simplify the calculations further, we employ a mean-field type approach by approximating $q^l_{-k_j^l}(x)$ by $q_s^l(x)$ and $s^l_{k_j^l}$ by its RMS value $\sqrt{q_s^l(x)}$. Note that we can also approximate $s^l_{k_j^l}$ by its mean value, giving the same qualitative results. This assumption simplifies Eqn. A.1 significantly, and we obtain

$$\left\langle q_s^l(x)|\boldsymbol{h}^{l-1}(x)\right\rangle = \frac{1}{N+1}\left[1+(N+1)\int dz\,du\,f(z)\,f(u)\,\phi\left(\sqrt{q_s^{l-1}(x)}(\sigma_w z+u)\right)^2\right].$$

To find the fixed point of the length map, we take a derivative wrt $q_s^{l-1}(x)$ to get the condition for stability of the fixed point $q^*(x)$. We denote the derivative by $\zeta_{q^*(x)}$. It separates the parameter space into two phases - a bounded phase when $\zeta_{q^*(x)} < 1$, and an unbounded phase when $\zeta_{q^*(x)} > 1$.

$$\begin{aligned}
\zeta_{q^*(x)} =& \frac{\partial q_s^l(x)}{\partial q_s^{l-1}(x)}\Big|_{q_s^{l-1}(x)=q^*(x)}\\
\zeta_{q^*(x)} =& \frac{\partial}{\partial q_s^{l-1}(x)}\int dz\,du\,f(z)\,f(u)\,\phi\left(\sqrt{q_s^{l-1}(x)}(\sigma_w z+u)\right)^2\\
\zeta_{q^*(x)} =& \frac{1}{\sqrt{q_s^{l-1}(x)}}\int dz\,du\,f(z)\,f(u)\,(\sigma_w z+u)\phi'\left(\sqrt{q_s^{l-1}(x)}(\sigma_w z+u)\right)\phi\left(\sqrt{q_s^{l-1}(x)}(\sigma_w z+u)\right)
\end{aligned}$$

$$\zeta_{q^*(x)} = \sigma_w^2\int dz\,du\,f(z)\,f(u)\,\left[\phi'\left(\sigma_w z+u\right)\right]^2+\sigma_w\int dz\,du\,f(z)\,f(u)\phi'\left(\sigma_w z+u\right)\phi\left(\sigma_w z+u\right) \tag{B.1}$$

where we have used the fact that for $a > 0$, $\phi(ax) = a\,\phi(x)$. On evaluating the integral numerically, we find that $\zeta_{q^*(x)} = 1$ when $\sigma_w^2 = 0.56$. On comparing with the numerical results in Fig. 3.6, we find that this approximated result underestimates the critical point at $\sigma_w^2 = 0.72$.

B.2 Stability of the fixed point for the correlation map for RAI

Under the assumption, $q_s^l(x) \to q^*(x)$, the correlation map has a fixed point $c_s^*(x_1,x_2) = 1$, and its stability can be calculated by the derivative of the correlation map evaluated at $c_s^{l-1}(x_1,x_2) = 1$. However, again, we get into the difficulties mentioned in the previous section, and we employ the same assumptions to arrive at a tractable equation for the correlation map

$$\left\langle c_s^l(x_1,x_2)|\boldsymbol{h}^{l-1}(x_1),\boldsymbol{h}^{l-1}(x_2)\right\rangle = \frac{1}{q_s^*(x)}\frac{1}{N+1}\left[1+N\int dz_1\, dz_2\, du\, f(z_1)f(z_2)\, f(u) \times \right.$$
$$\left. \times\ \phi\left(\sqrt{q_s^*(x)}(\sigma_w\, z_1+u)\right)\phi\left(\sqrt{q_s^*(x)}\left[c_s^{l-1}(x_1,x_2)\,\sigma_w\, z_1+\sqrt{1-(c_s^{l-1}(x_1,x_2))^2}\,\sigma_w\, z_2+u\right]\right)\right].$$

Next, we take a derivative to get the condition for the stability of the fixed point $c_h^*(x_1,x_2)=1$

$$\begin{aligned}
\chi_1 =& \frac{\partial c_h^l(x_1,x_2)}{\partial c_h^{l-1}(x_1,x_2)}\Big|_{c_h^{l-1}(x_1,x_2)=1}\\
\chi_1 =& \frac{1}{q_h^*(x)}\frac{\partial}{\partial c_h^{l-1}(x_1,x_2)}\int dz_1dz_2du f(z_1)f(z_2)f(u)\phi\left(\sqrt{q_s^*(x)}(\sigma_w\, z_1+u)\right)\times\\
&\times\phi\left(\sqrt{q_s^*(x)}\left[c_s^{l-1}(x_1,x_2)\,\sigma_w\, z_1+\sqrt{1-(c_s^{l-1}(x_1,x_2))^2}\,\sigma_w\, z_2+u\right]\right)\Big|_{c_h^{l-1}(x_1,x_2)=1}
\end{aligned}$$

$$\chi_1=\sigma_w^2\int dz\, du\, f(z)\, f(u)\ \left[\phi'(\sigma_w z+u)\right]^2. \tag{B.2}$$

The above equation is the same as the first term we obtained in the condition for the stability of the length map (Eqn. B.1). Finally, we obtain a critical value of $\sigma_w^2=1.41$ by solving for $\chi_1=1$. We observe that the critical point for the length is smaller than the critical point of the correlation coefficient, and from our experience with ReLU networks with correlated weights, we expect RAI to have an ordered phase only, which is confirmed by experimental results (see Fig. 3.6).

Appendix C

Derivation of length and correlation map for RAAI

C.1 Derivation for the length map for RAAI and the stability condition

For given $\boldsymbol{h}^{l-1}(x)$, we can view $h_j^l(x)$ as

$$h_j^l(x) = \left(\sigma_w \sqrt{\tilde{q}^{l-1}(x)}\, z + s^{l-1}_{-k_j^{l-1}}(x)\, u\right),$$

where $\tilde{q}^{l-1} = q^{l-1}_{-k_j^{l-1}}(x) - \frac{k}{1+k}\left(m^{l-1}_{-k_j^{l-1}}(x)\right)^2$. Here $m^{l-1}_{-k_j^{l-1}}(x)$ is the mean value of $s^{l-1}_{-k_j^{l-1}}$ averaged over nodes in the previous layer. We can use the relation $m_s^l(x)^2 = q_s^l(x)/\pi$ and re-define σ_w as

$$\tilde{\sigma}^2 = \sigma_w^2\left(1 - \frac{k}{1+k}\frac{1}{\pi}\right),$$

which yields,

$$h_j^l(x) = \left(\tilde{\sigma} \sqrt{q_s^{l-1}(x)}\, z + s^{l-1}_{-k_j^{l-1}}(x)\, u \right),$$

Now, the entire analysis goes through as Section B.1, just with a re-definition of the variance. Now, we can read off the stability condition for the fixed point of the length map

$$\zeta_{q^*(x)} = \tilde{\sigma}_w^2 \int dz\, du\, f(z)\, f(u)\ \left[\phi'(\tilde{\sigma} z + u)\right]^2 + \tilde{\sigma} \int dz\, du\, f(z)\, f(u) \phi'(\tilde{\sigma} z + u)\, \phi(\tilde{\sigma} z + u) \quad \text{(C.1)}$$

On solving the equations numerically, we observe that the length is bounded when $\sigma_w^2 < 1.75$, which overestimates the numerical value observed in experiments (see Fig. 3.7).

C.2 Derivation for the correlation map for RAAI and the stability condition

The derivation for the correlation map for RAAI can be done similar to RAI, with a key difference of the covariance matrix. The covariance matrix is given by

$$\Sigma_{k_j^l}^{l-1}(x_1,x_2) = \begin{bmatrix} q^{l-1}_{-k_j^{l-1}}(x_1) - \frac{k}{1+k}\left(m^{l-1}_{-k_j^{l-1}}(x_1)\right)^2 & q^{l-1}_{-k_j^{l-1}}(x_1,x_2) - \frac{k}{1+k} m^{l-1}_{-k_j^{l-1}}(x_1) m^{l}_{-k_j^{l-1}}(x_2) \\ q^{l-1}_{-k_j^{l-1}}(x_1,x_2) - \frac{k}{1+k} m^{l-1}_{-k_j^{l-1}}(x_1) m^{l}_{-k_j^{l-1}}(x_2) & q^{l-1}_{-k_j^{l-1}}(x_2) - \frac{k}{1+k}\left(m^{l-1}_{-k_j^{l-1}}(x_2)\right)^2 \end{bmatrix}.$$

Again, $c^*(x_1,x_2) = 1$ is a fixed point of the recursive map. On performing approximations as in Section B.2, we find that the stability condition is again given by Eqn. B.2, which gives $\sigma_w^2 = 1.41$ as the phase transition boundary. Comparing it with experiments (Fig. 3.7), we find that it overestimates the critical value.

Note that instead of approximating $s^{l-1}_{k_j^{l-1}}$ by its RMS value $\sqrt{q_s^{l-1}(x)}$, we can also approximate it by its mean value. In this case, we find that we overestimate all the boundaries. In Table C.1, we compare the boundaries predicted by the RMS and mean approximations.

Approximation	$(\sigma_w^2)_q(RAI)$	$(\sigma_w^2)_c(RAAI)$	$(\sigma_w^2)_q(RAAI)$
RMS	0.57	1.41	1.75
Mean	0.85	1.46	1.89

Table C.1: A comparison of the decision boundaries obtained by approximating $s^{l-1}_{k^{l-1}_j}$ by its RMS and mean value. The RMS approximation underestimates the length boundary for RAI, whereas it overestimates both the phase boundaries for RAAI. On the other hand, the mean approximation overestimates the phase boundaries for both RAI and RAAI.

Appendix D

Code to generate weights drawn from RAAI distribution

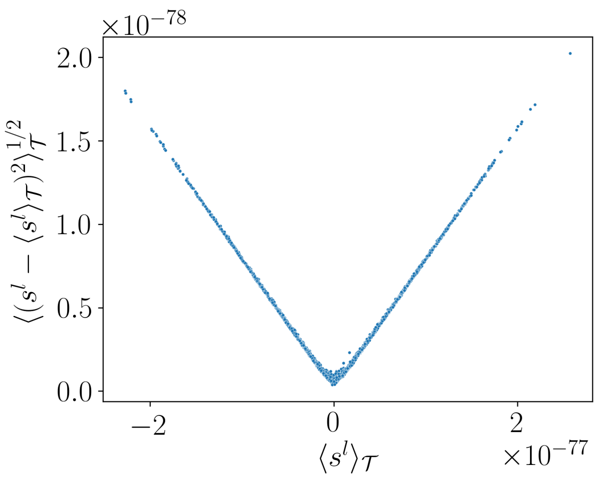
```
import numpy as np
def RAAI(fan_in, fan_out, k = 100, variance_weights = 0.9):
    """Randomized Asymmetric Anti-correlated Initializer (RAAI)
    Arguments:
    fan_in -- the number of neurons in the previous layer
    fan_out -- the number of neurons in the next layer
    k -- correlation strength for the Gaussian weights
    variance_weights -- variance of the weights
    Returns:    W, b -- weight and bias matrices with shape(fan_in,
   fan_out), and (fan_out, ) """
    corr = k/(1+k)
    mean = np.zeros(fan_in + 1)
    J = np.ones((fan_in + 1, fan_in + 1))
    cov = (np.identity(fan_in + 1) - J*(corr/(fan_in +1)) )*
   variance_weights/fan_in
    P = np.random.multivariate_normal(mean = mean, cov = cov, size = (
   fan_out))
    for j in range(P.shape[0]):
        k = np.random.randint(0, high = fan_in + 1)
        P[j, k] = np.random.beta(2, 1)
    W = P[:, :-1].T
    b = P[:, -1]
    return  W.astype(np.float32), b.astype(np.float32)
```

Appendix E

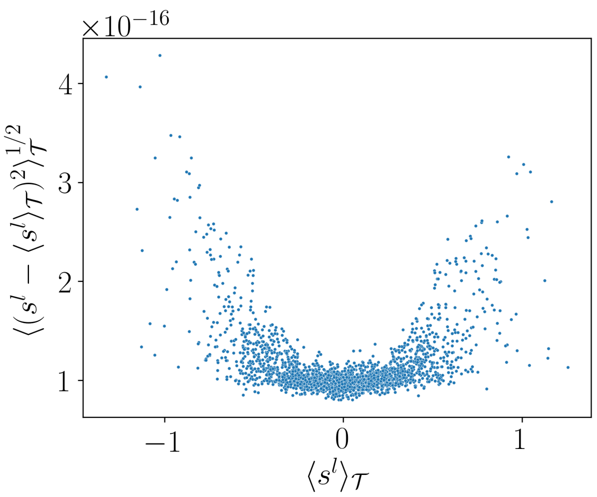
Properties of ordered and chaotic phase

In this section, we attempt to explore the distribution of values a node attains. As the pre-activations are Gaussian distributed, we plot the mean against the standard deviation over the training set for the pre-activations for different nodes.

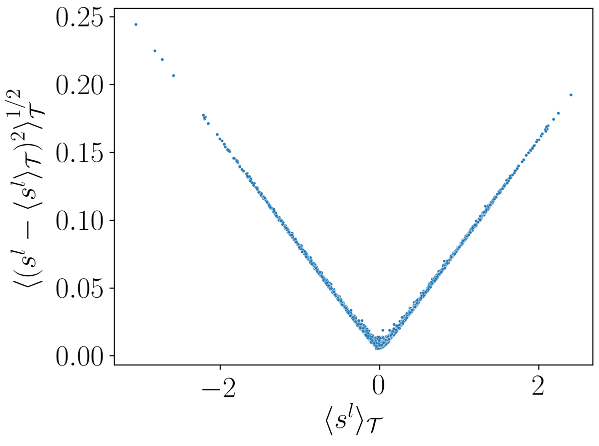
We observe that for ReLU networks with uncorrelated weights, which only have an ordered phase, the mean value and the standard deviation of the pre-activations for a node are related, whereas, for ReLU networks with anti-correlated weights, we see two different cases; in the ordered phase, the mean versus standard deivation show some correlation, whereas, in the chaotic phase and the phase transition boundary they seem to be uncorrelated.



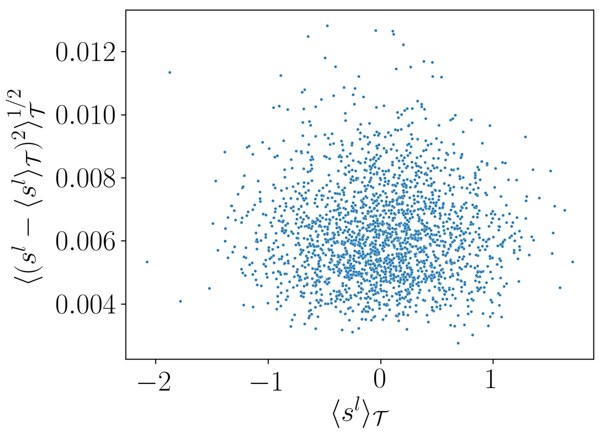
(a) Uncorrelated initialization with $\sigma_w^2 = 1, \sigma_b^2 = 0$



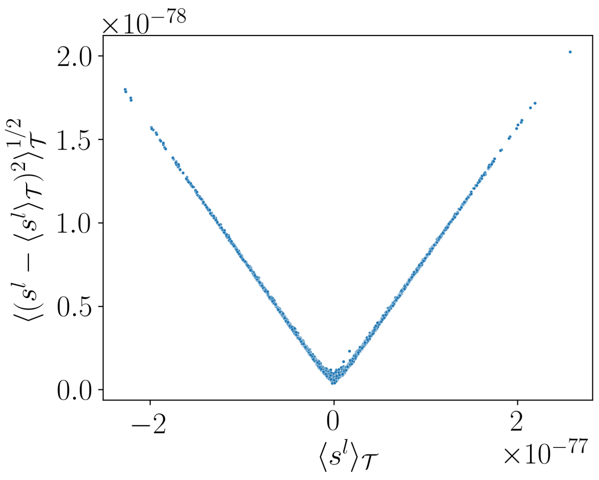
(b) ACI with $\sigma_w^2 = 1, \sigma_b^2 = 0.1$



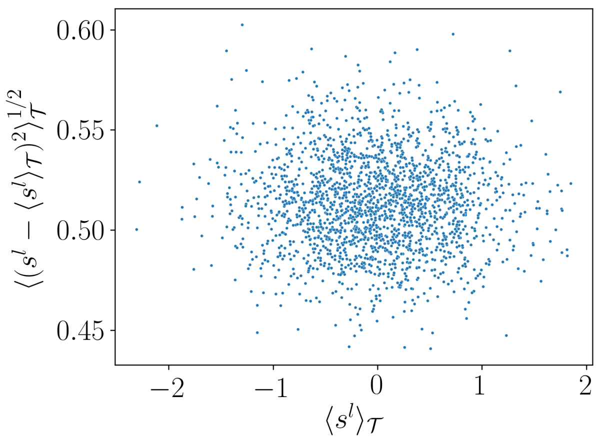
(c) Uncorrelated initialization with $\sigma_w^2 = 2, \sigma_b^2 = 0$



(d) ACI with $\sigma_w^2 = 2, \sigma_b^2 = 0.1$



(e) Uncorrelated initialization with $\sigma_w^2 = 3, \sigma_b^2 = 0$



(f) ACI with $\sigma_w^2 = 2.5, \sigma_b^2 = 0.1$

Figure E.1: Mean versus standard deviation of the pre-activation at various nodes