國立臺灣大學電機資訊學院電信工程學研究所
碩士論文
Graduate Institute of Communication Engineering
College of Electrical Engineering and Computer Science
National Taiwan University
Master Thesis

神經元消失：造成深層神經網路難以訓練的另一種現象

Vanishing Nodes: Another Phenomena That Makes Training Deep Neural Networks Difficult

張文于
Wen-Yu Chang

指導教授：林宗男
Advisor: Tsung-Nan Lin

中華民國 108 年 7 月
July, 2019

摘要

梯度爆炸/消失，一直被認為是訓練深層神經網路的一大挑戰。在這篇論文裡，我們發現一種被稱為「神經元消失 (Vanishing Nodes)」的新現象同樣也會使訓練更加困難。當神經網路的深度增加，神經元彼此之間的會呈現高度相關。這種行為會導致神經元之間的相似程度提高。也就是隨著神經網路變深，網路內的神經元冗餘程度會提高。我們把這個問題稱為「神經元消失 (Vanishing Nodes)」。可以藉由神經網路的相關參數來對神經元消失的程度做推算；結果可以得出神經元消失的程度與網路深度成正比、與網路寬度成反比。從數值分析的結果呈現出：在反向傳播算法的訓練下，神經元消失的現象會變得更明顯。我們也提出：神經元消失是除了梯度爆炸/消失以外，訓練深層神經網路的另一道難關。

關鍵字： 深度學習, 梯度消失, 機器學習理論

Abstract

It is well known that the problem of vanishing/exploding gradients creates a challenge when training deep networks. In this paper, we show another phenomenon, called *vanishing nodes*, that also increases the difficulty of training deep neural networks. As the depth of a neural network increases, the network's hidden nodes show more highly correlated behavior. This correlated behavior results in great similarity between these nodes. The redundancy of hidden nodes thus increases as the network becomes deeper. We call this problem "*Vanishing Nodes*." This behavior of vanishing nodes can be characterized quantitatively by the network parameters, which is shown analytically to be proportional to the network depth and inversely proportional to the network width. The numerical results suggest that the degree of vanishing nodes will become more evident during back-propagation training. Finally, we show that vanishing/exploding gradients and vanishing nodes are two different challenges that increase the difficulty of training deep neural networks.

Keywords: Deep learning, Vanishing gradient, Learning theory

Contents

摘要 iii

Abstract v

1 Introduction 1

2 Related Work 3

2.1 Difficulties in training deep nerual networks 3

2.2 Representation power of deep neural network 3

3 Vanishing Nodes: correlation between hidden nodes 5

3.1 Vanishing Node Indicator 6

3.2 Impacts of back-propagation 10

3.3 Representation power vanishes as the network goes deeper 11

3.4 The effect of the orthogonal weight matrices to the representation power . 11

3.5 Representation power of residual-like architectures 11

4 Variance propagation of deep neural networks 17

4.1 Comparison with exploding/vanishing gradients 17

4.2 Norm-preserving weight initialization 19

4.3 Batch normalization results in exploding gradients 19

4.4 T-normalization: A novel normalization method for deep networks 19

5 Experiments 21

5.1 Probability of failed training 21

5.2 Analyses of failed training 26

6 Conclusion **29**

Bibliography **31**

List of Figures

3.1 Scatter plots of output layer nodes. 8

3.2 The results of VNI R_{sq} with respect to network depth L. 12

3.3 The magnitudes of correlation coefficient ρ_{ij} between output nodes. . . . 13

3.4 The dynamics of VNI R_{sq} of the output layer. 13

3.5 The averages of squared correlation coefficients ρ_{ij}^2 over 50 runs. 14

3.6 The initial R_{sq} increases to 1 as the network goes deeper. 15

3.7 The initial R_{sq} increases to 1 as the network goes deeper. 16

5.1 Probability of successful training for the SGD optimizer. 22

5.2 Probability of successful training for the SGD + Momentum optimizer. . . 23

5.3 Probability of successful training for the Adam optimizer. 24

5.4 Probability of successful training for the RMSProp optimizer. 25

5.5 Box and whisker plot of $\sigma_w^2\mu_1$ for networks with successful training. . . . 27

5.6 Box and whisker plot of $\sigma_w^2\mu_1$ for networks with failed training. 27

5.7 Histogram of R_{sq} of failed/successful networks. 28

List of Tables

5.1 The detailed numbers of successful and failed runs. 28

Chapter 1

Introduction

Deep neural networks (DNN) have succeeded in various fields, including computer vision [17], speech recognition [16], machine translation [33], medical analysis [29] and human games [27]. Some results are comparable to or even better than those of human experts.

State-of-the-art methods in many tasks have recently used increasingly *deep* neural network architectures. The performance has improved as networks have been made *deeper*. For example, some of the best-performing models [14, 15] in computer vision have included hundreds of layers.

Moreover, recent studies have found that as the depth of a neural network increases, problems such as vanishing or exploding gradients make the training process more challenging. [7, 13] investigated this problem deeply and suggested that initializing weights in appropriate scales can prevent gradients from vanishing or exploding exponentially. [22, 26] also studied how vanishing/exploding gradients arise via *mean field theory* and provided a solid theoretical discriminant to determine whether the propagation of gradients is vanishing/exploding.

Inspired by previous studies, we investigated the correlation between hidden nodes and discovered that a phenomenon that we call *vanishing nodes* can also affect the capability of a neural network. In general, the hidden nodes of a neural network become highly correlated as the network becomes deeper. The correlation between nodes implies the similarity between them, and high degree of similarity between nodes produces redundancy. Because a sufficient number of effective nodes is needed to approximate an

arbitrary function, the redundancy of nodes in hidden layers may debilitate the representation capability of the entire network. Thus, as the depth of the network increases, the redundancy of hidden nodes may increase and hence affect the network's trainability. We name this phenomena as "*Vanishing Nodes*."

We propose a *Vanishing Node Indicator (VNI)*, which is the weighted average of squared correlation coefficients, as the quantitative metric for vanishing nodes. VNI can be theoretically approximated via the results on the spectral density of the end-to-end Jacobian. The approximation of VNI depends on the network parameters, including the width, the depth, the distribution of weights, and the activation functions, and it is shown to be simply proportional to the network depth and inversely proportional to the network width.

In addition, the numerical results show that back-propagation training also intensifies the correlations of hidden nodes when we consider a deep network. We find that although we use a relatively large network width, the correlations of hidden nodes may still increase during the training process.

Finally, we show that vanishing/exploding gradients and vanishing nodes are two different problems, so that the two problems may arise from specific conditions. The experimental results show that the likelihood of failed training increases as the depth of the network increases. The training will become much more difficult due to lack of network representation capability.

This paper is organized as follows: some related works are discussed in Section 2. The vanishing nodes phenomenon is introduced in Section 3. Theoretical analysis and a quantitative metric are reported in Section 3. Section 4 compares the vanishing nodes with vanishing/exploding gradients. Section 5 reports the experimental results and Section 6 gives our conclusions.

Chapter 2

Related Work

2.1 Difficulties in training deep nerual networks

Problems in the training of deep neural networks have been encountered in several studies. For example, [7, 13] investigated vanishing/exploding gradient propagation and gave weight initialization methods as the solution. [10] suggested that vanishing/exploding gradients might relate to the sum of the reciprocals of the hidden layer widths. [8, 5] stated that saddle points are more likely than local minima to be a problem for training deep neural networks. [12, 28, 14] exposed the *degradation* problem: the performance of a deep neural network degrades as the depth increases.

Dynamical isometry is one of the conditions that make ultra-deep network training more feasible. [25] reported dynamical isometry to theoretically ensure depth-independent learning speed. [20, 21] suggested several ways to achieve dynamical isometry for various settings of network architecture, and [34, 2] practically trained ultra-deep networks in various tasks.

2.2 Representation power of deep neural network

Representation power has been surveyed in many previous works. According to the ”universal approximation theorem” proved by [4], a single hidden layer with a finite number of neurons can approximate any continuous function on compact subsets. However, [31]

states that the network depth of neural networks governs the representation power and the training performance. Theoretically, [22, 23, 11, 24] claim the expressive complexity of a network grows exponentially with its depth but not its width. For ReLU networks, [1, 30, 9] show that the minimal number of nodes to aprroximate any continuous function can be reduced if the depth of the network is larger.

The correlation between the nodes of hidden layers within a deep neural network is our main focus. As we know, the correlation between nodes implies the similarity between them, and high degree of similarity between nodes produces redundancy, hence reduce the representation power of the network. Several kinds of correlations have been discussed in the literature. In this work, we proposed a different problem related to the correlation between two nodes in a hidden layer. [26] surveyed the propagation of the correlation between two different inputs after several layers. [19, 32] suggested that the input features must be whitened (i.e., zero-mean, unit variances and uncorrelated) to achieve a faster training speed.

Chapter 3

Vanishing Nodes: correlation between hidden nodes

In this section, the correlation of hidden-layer neurons is investigated. If a pair of neurons is highly correlated (for example, the correlation coefficient is equal to $+1$ or -1), one of the neurons becomes redundant. Great similarity between nodes may reduce the effective number of neurons within a network. In some cases, the correlation of hidden nodes may disable the entire network. This phenomenon is called *Vanishing Nodes*.

First, consider a deep feed-forward neural network with depth L. For simplicity of analysis, we assume all layers have the same width N. The weight matrix of layer l is $\mathbf{W}_l \in \mathbb{R}^{N\times N}$, the bias of layer l is $\mathbf{b}_l \in \mathbb{R}^N$ (a column vector), and the common activation function of all layers is $\phi(\cdot) : \mathbb{R} \to \mathbb{R}$. The input of the network is $\mathbf{x}_0$, and the nodes at output layer L denote $\mathbf{x}_L$. The pre-activation of layer l is $\mathbf{h}_l \in \mathbb{R}^N$ (a column vector), and the post-activation of layer l is $\mathbf{x}_l \in \mathbb{R}^N$ (a column vector). That is, $\forall l \in \{1, ..., L\}$,

$$\mathbf{h}_l = \mathbf{W}_l\mathbf{x}_{l-1} + \mathbf{b}_l, \quad \mathbf{x}_l = \phi(\mathbf{h}_l). \tag{3.1}$$

The variance of node i is defined as $\sigma_i^2 \triangleq \mathbb{E}_{\mathbf{x}_0}[(x_{l(i)} - \overline{x_{l(i)}})^2]$, and the squared correlation coefficient (ρ_{ij}^2) between nodes i and j can be computed as

$$\rho_{ij}^2 \triangleq \frac{\mathbb{E}_{\mathbf{x}_0}[(x_{l(i)} - \overline{x_{l(i)}})(x_{l(j)} - \overline{x_{l(j)}})]^2}{\mathbb{E}_{\mathbf{x}_0}[(x_{l(i)} - \overline{x_{l(i)}})^2]\mathbb{E}_{\mathbf{x}_0}[(x_{l(j)} - \overline{x_{l(j)}})^2]}, \tag{3.2}$$

where ρ_{ij}^2 ranges from 0 to 1. Nodes $x_{l(i)}$ and $x_{l(j)}$ are highly correlated only if the magnitude of the correlation coefficient between two nodes ρ_{ij} is nearly 1. ρ_{ij}^2 indicates the magnitude of similarity between node i and node j. If ρ_{ij} is close to $+1$ or -1, then node i can be approximated in a linear fashion by node j. Great similarity indicates redundancy. If nodes of hidden layers exhibit great similarity, the effective number of nodes will be much lower than the original network width. Therefore, we call this phenomena *Vanishing Node Problem*.

In the following section, we propose a metric to measure the quantitative property of vanishing nodes for a deep feed-forward neural network. Theoretical analysis of the metric indicates that the quantitative property of vanishing nodes is proportional to the network depth and inversely proportional to the network width. The quantity is shown analytically to depend on the statistical property of weights and the nonlinear activation function.

3.1 Vanishing Node Indicator

Consider the network architecture defined in eqn. (3.1). In addition, the following assumptions are made: (1) The input $\mathbf{x}_0$ is zero-mean, and the features in $\mathbf{x}_0$ are independent and identically distributed. (2) All weight matrices $\mathbf{W}_l$ in each layer are initialized from the same distribution with variance σ_w^2/N. (3) All the bias vectors $\mathbf{b}_l$ in each layer are initialized to zero.

The input-output Jacobian matrix $\mathbf{J} \in \mathbb{R}^{N\times N}$ is defined as the first-order partial derivative of the output layer with respect to the input layer, which can be rewritten as $\frac{\partial \mathbf{x}_L}{\partial \mathbf{x}_0} = \prod_{l=1}^{L} \mathbf{D}_l \mathbf{W}_l$, where $\mathbf{D}_l \triangleq diag(\phi'(\mathbf{h}_l))$ is the derivative of point-wise activation function ϕ at layer l. To conduct a similar analysis as [25], consider the first-order forward approximation: $\mathbf{x}_L - \overline{\mathbf{x}_L} \approx \mathbf{J}\mathbf{x}_0$. Therefore, the covariance matrix of the nodes ($\mathbf{C} \in \mathbb{R}^{N\times N}$) at the output layer can be computed as

$$\mathbf{C} \triangleq \mathbb{E}_{\mathbf{x}_0}[(\mathbf{x}_L - \overline{\mathbf{x}_L})(\mathbf{x}_L - \overline{\mathbf{x}_L})^T] \approx \mathbb{E}_{\mathbf{x}_0}[(\mathbf{J}\mathbf{x}_0)(\mathbf{J}\mathbf{x}_0)^T] = \mathbf{J}\mathbb{E}_{\mathbf{x}_0}[\mathbf{x}_0\mathbf{x}_0^T]\mathbf{J}^T = \sigma_x^2 \mathbf{J}\mathbf{J}^T, \quad (3.3)$$

where σ_x^2 is the common variance of features in $\mathbf{x}_0$, and the expected values are calculated with respect to the input $\mathbf{x}_0$. For notational simplicity, we omit the subscript $\mathbf{x}_0$ of the expectations in the following equations. It can be easily derived that the squared covariance of nodes i and j is equal to the product of the squared correlation coefficient and the two variances. That is, $[C_{(ij)}]^2 = \rho_{ij}^2 \sigma_i^2 \sigma_j^2$.

In this paper, we propose the *Vanishing Node Indicator (VNI)* R_{sq} to quantitatively characterize the degree of vanishing nodes for a given network architecture. It is defined as follows:

$$R_{sq} \triangleq \frac{\sum_{i=1}^{N} \sum_{j=1}^{N} \rho_{ij}^2 \sigma_i^2 \sigma_j^2}{\sum_{i=1}^{N} \sum_{j=1}^{N} \sigma_i^2 \sigma_j^2}. \tag{3.4}$$

VNI calculates the weighted average of the squared correlation coefficients ρ_{ij}^2 between output layer nodes with non-negative weights $\sigma_i^2 \sigma_j^2$. Basically, VNI R_{sq}, which ranges from $1/N$ to 1, summarizes the similarity of the nodes at the output layer. If all nodes are independent of each other, the correlation coefficients ρ_{ij} will be 0 (if $i \neq j$) or 1 (if $i = j$) and R_{sq} will become the minimum value of $1/N$. Otherwise, if all of the output nodes are highly correlated, then all squared correlation coefficients ρ_{ij}^2 will be nearly 1, and therefore R_{sq} will reach the maximum value of 1. Note that the weights $\sigma_i^2 \sigma_j^2$ in the weighted average can be interpreted as the importance of the output-layer nodes i and j. If all of the output layer nodes have equal variances, VNI R_{sq} is simply reduced to the average of the squared correlation coefficients ρ_{ij}^2.

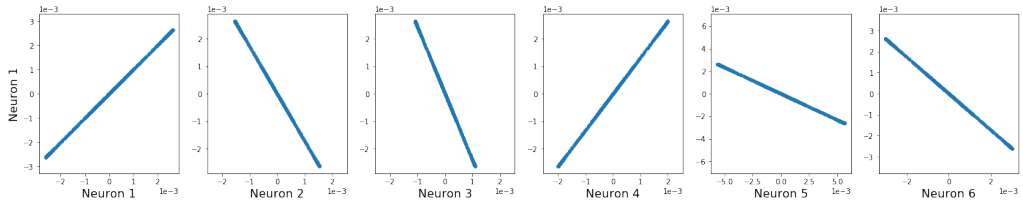
With the covariance matrix defined in eqn. (3.3) and the formulas for matrix traces, VNI R_{sq} can be expressed as the formula of the covariance matrix as

$$\begin{aligned} R_{sq} &= \frac{\sum_{i=1}^{N} \sum_{j=1}^{N} \mathbb{E}_{\mathbf{x}_0}[(x_{L(i)} - \overline{x_{L(i)}})(x_{L(j)} - \overline{x_{L(j)}})]^2}{\sum_{i=1}^{N} \sum_{j=1}^{N} \mathbb{E}_{\mathbf{x}_0}[(x_{L(i)} - \overline{x_{L(i)}})^2] \mathbb{E}[(x_{L(j)} - \overline{x_{L(j)}})^2]} \\ &= \frac{\sum_{i=1}^{N} \sum_{j=1}^{N} [C_{(ij)}]^2}{\sum_{i=1}^{N} \sum_{j=1}^{N} C_{(ii)} C_{(jj)}} = \frac{tr(\mathbf{C}\mathbf{C}^T)}{tr(\mathbf{C})^2}, \end{aligned} \tag{3.5}$$

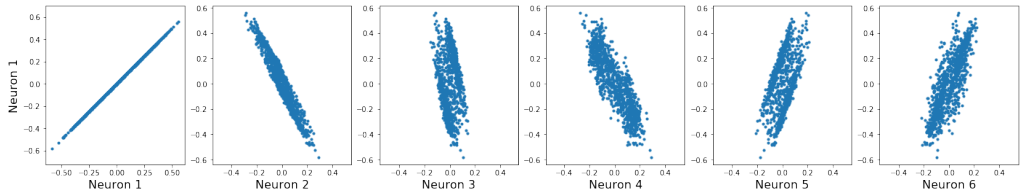
where $tr(\cdot)$ is the matrix trace operation.

From eqn. (3.3), substituting $\sigma_x^2 \mathbf{J}\mathbf{J}^T$ for $\mathbf{C}$ in eqn. (3.5), and noting that $tr(\mathbf{A}^k)$ is equal

(a) Network width $N = 6$



(b) Network width $N = 50$



(c) Network width $N = 100$

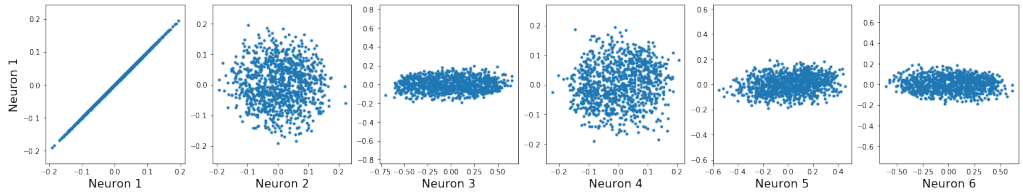


Figure 3.1: To show the output layer correlation, we plot the scatter plots with 1000 data points of 6 random sampled output layer nodes of the network with network depth $L = 100$, $Hard\text{-}Tanh$ activation and scaled-Gaussian weight initialization. The network width $N = 6, 50, 100$ from top to bottom respectively. We can see that correlations are much higher when the network width N is small.

to the sum of eigenvalues to the k-th power of symmetric matrix $\mathbf{A}$ [6], an approximation of R_{sq} can be obtained:

$$R_{sq} \approx \frac{tr(\mathbf{J}\mathbf{J}^T\mathbf{J}\mathbf{J}^T)}{tr(\mathbf{J}\mathbf{J}^T)^2} = \frac{\sum_{k=1}^{N}\lambda_k^2}{(\sum_{k=1}^{N}\lambda_k)^2} = \frac{N \cdot m_2}{(N \cdot m_1)^2} = \frac{m_2}{N m_1^2}, \tag{3.6}$$

where λ_k is the k-th eigenvalue of $\mathbf{J}\mathbf{J}^T$, and m_i is the i-th moment of eigenvalues of $\mathbf{J}\mathbf{J}^T$.

In eqn. (3.6), we show that R_{sq} is related to the expected moments of the eigenvalues of $\mathbf{J}\mathbf{J}^T$. Because the moments of the eigenvalues of $\mathbf{J}\mathbf{J}^T$ have been analyzed in previous studies [21], we can leverage the recent results by [21]: $m_1 = (\sigma_w^2\mu_1)^L$, and $m_2 = (\sigma_w^2\mu_1)^{2L}L\big(\frac{\mu_2}{\mu_1^2} + \frac{1}{L} - 1 - s_1\big)$, where σ_w^2/N is the variance of the initial weight matrices, s_1 is the first moment of the series expansion of the S-transform associated with the weight matrices, and μ_k are the k-th moments of series expansion of the moment generating function associated with activation functions. If we insert the expressions of m_1 and m_2 into eqn. (3.6), we can obtain an approximation of the expected VNI:

$$R_{sq} \approx \frac{L}{N}\Big(\frac{\mu_2}{\mu_1^2} + \frac{1}{L} - 1 - s_1\Big) = \frac{1}{N} + \frac{L}{N}\Big(\frac{\mu_2}{\mu_1^2} - 1 - s_1\Big), \tag{3.7}$$

which shows that VNI is determined by the depth L, the width N, the moments of the activation functions μ_k and the statistical property of weights s_1. Because R_{sq} ranges from $1/N$ to 1, the approximation in eqn. (3.7) is more accurate when $N >> L$. Moreover, it can be easily seen that the correlation is inversely proportional to the network width N, and proportional to the network depth L.

To evaluate the accuracy of eqn. (3.7) with respect to the original definition in eqn. (3.4), we design the following experiments. A network width, $N \in \{200, 500\}$, is set. The network depth L is adjusted from 10 to 100 with the Hard-Tanh activation function. One thousand data points with the distribution $\mathbf{x}_0 \sim Gaussian(\mu_x = 0, \sigma_x^2 = 0.1)$ and 50,000 training images in MNIST dataset [18] are fed into the network. In each network architecture, the weights are initialized with scaled-Gaussian distribution [7] of various random seeds for 100 runs. The R_{sq} calculated from eqn. (3.4) is then recorded to compute the mean and the standard deviation with respect to various network depths L. The results

are shown in Figure 3.2 as the blue and green lines denoted "Simulation i.i.d. inputs" and "Simulation MNIST dataset." The red line denoted as "Theoretical" is the result calculated from eqn. (3.7). This experiment demonstrates that VNI expressed in terms of the network parameters in eqn. (3.7) is very close to the original definition in eqn. (3.4). Similar results are obtained with different activations (e.g., Linear, ReLU) and different weight initialization (e.g., scaled uniform distribution).

Figure 3.3 plots the squared correlation coefficients between output nodes, which are evaluated with 50,000 training images in the MNIST dataset [18] for various network architectures. White indicates no correlation, and black means that $\rho_{ij}^2 = 1$. Figure 3.3 (a) plots the squared correlation coefficients for four architectures with the same network width ($N = 200$) at different depths (5, 50, 300, and 1000). Figure 3.3 (b) shows the architectures with the same depth ($L = 100$) and different widths (5, 50, 200, 1000). This shows that the vanishing node phenomenon becomes evident with respect to the depth and inversely proportional to the width.

3.2 Impacts of back-propagation

In Section 3.1, we showed that the correlation of a network will increase as the depth L increases; in this section, we exploit the manner in which the back-propagation training process will influence the network correlation by the following experiments.

First, the same architecture defined in eqn. (3.1), with $L = 100$, $N = 500$, tanh activation, and scaled Gaussian initialization [7], is used. The network is then trained on the MNIST dataset [18] and optimized with stochastic gradient descent (SGD) with a batch size of 100. The network is trained with three different learning rates for different seeds to initialize the weights for 20 runs. We then record the quartiles of VNI (R_{sq}) with respect to the training epochs, as shown in Figure 3.4.

The boundaries of the colored areas represent the first and third quartiles (i.e., the 25th and 75th percentiles), and the line represents the second quartile (i.e., the median) of R_{sq} over 20 trials. It shows that in some cases, VNI increases to 1 during the training process, otherwise VNI grows larger initially, and then decreases to a value which is larger than the

initial VNI. Severe intensification of VNI may occur, as shown by the blue line, which is trained at the learning rate of 10^{-2}. Moreover, we observe that training will become much more difficult due to a lack of network representation capability as VNI R_{sq} approaches 1. Further discussion is provided in Section 5 to investigate the impact of VNI by various training parameters.

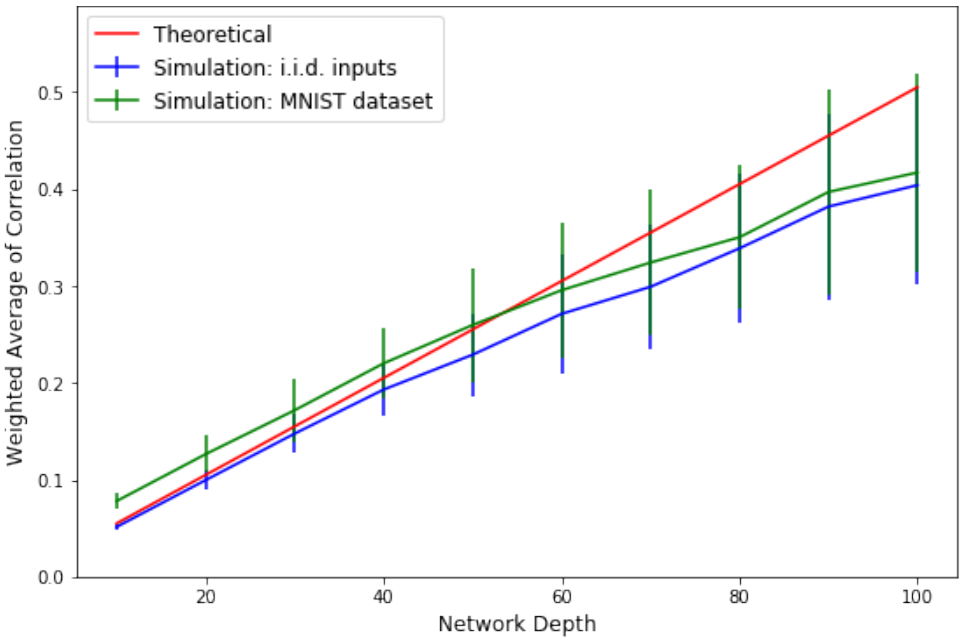
In Figure 3.5, we present the pairwise averaged squared correlation coefficients ρ_{ij}^2 via its color. The architecture is defined with $L = 100$, $N = 500$, tanh activation and scaled Gaussian initialization. Random samples from the distribution $\mathbf{x}_0 \sim Gaussian(\mu_x = 0, \sigma_x^2 = 0.1)$ with batch size 1000 are used as the input data, and the same distribution is used as the output gradients. The network is trained by SGD optimization with learning rate $= 10^{-2}$. The darker pixels represent higher correlations. Note that in Figure 3.5a, the color of layer 100 is bright because the network width $N = 500$ is large relative to the network depth $L = 100$.

3.3 Representation power vanishes as the network goes deeper

3.4 The effect of the orthogonal weight matrices to the representation power

3.5 Representation power of residual-like architectures

(a) Network width $N = 200$



(b) Network width $N = 500$

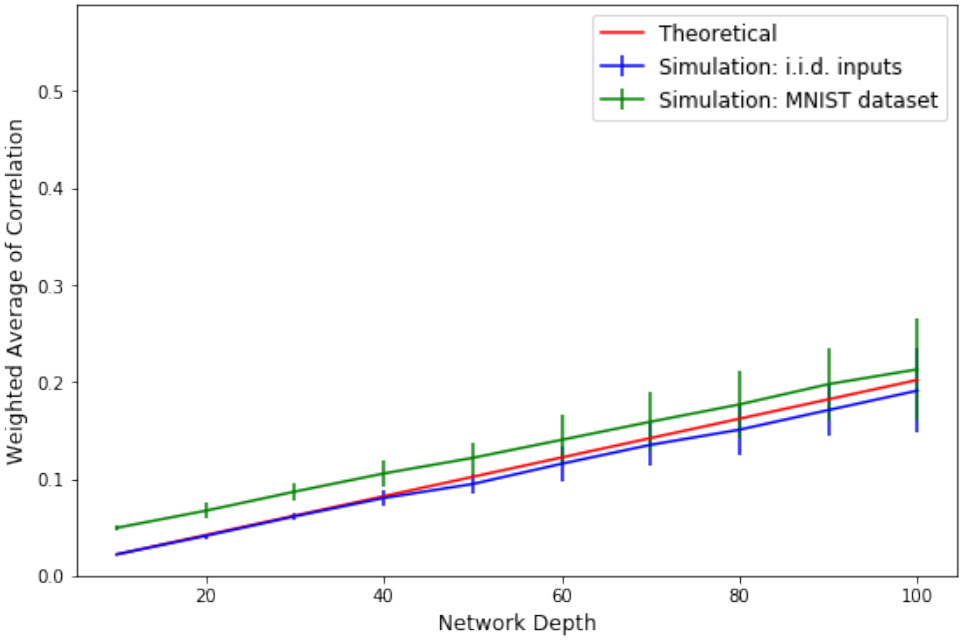
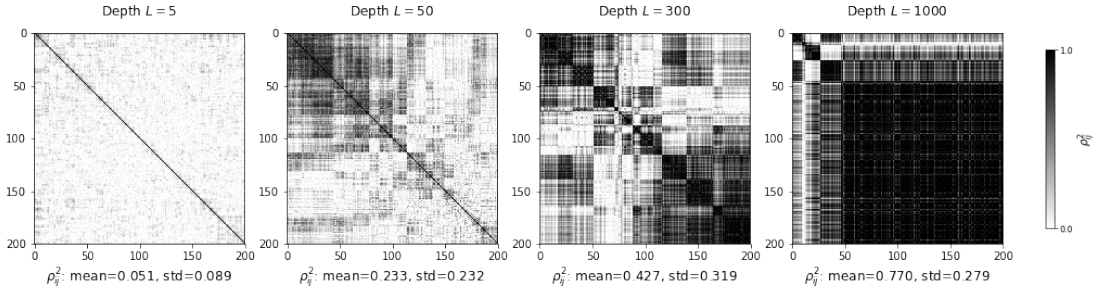


Figure 3.2: The results of VNI R_{sq} with respect to network depth L for the network width 200 and 500. The red line is calculated from eqn. (3.7), the blue line is computed from eqn. (3.4) with the input data of zero mean and i.i.d input data, and the green line is computed from eqn. (3.4) with MNIST data. The VNI R_{sq} expressed in eqn. (3.7) is very close to the original definition in eqn. (3.4).

(a) Network width $N = 200$



(b) Network depth $L = 100$

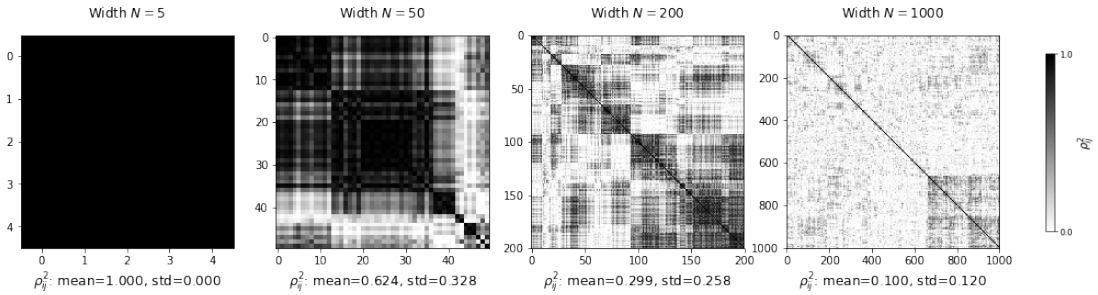


Figure 3.3: The magnitudes of correlation coefficient ρ_{ij} between output nodes. The black color means $\rho_{ij}^2 = 1$ while the white color indicates $\rho_{ij}^2 = 0$. The top row shows that the correlation is positive related to the network depth L, and the bottom row presents that the correlation is negatively related to the network width N. Note that we rearrange the node index to cluster the correlated nodes.

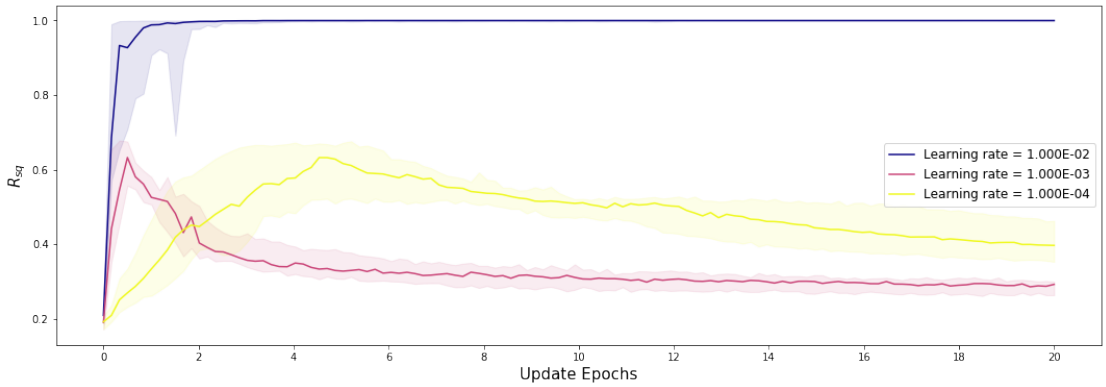
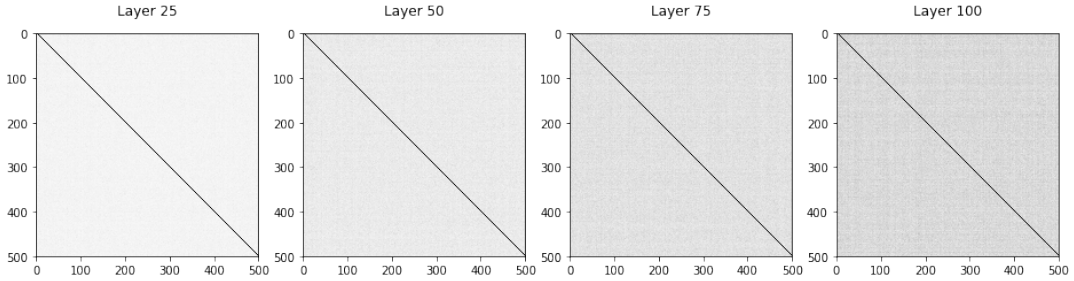
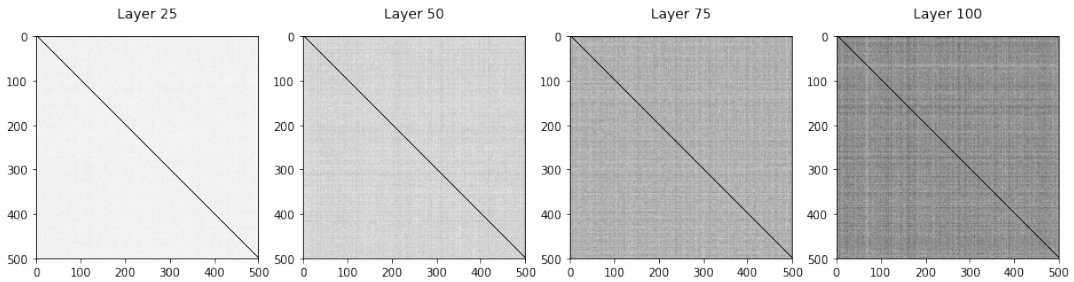


Figure 3.4: The dynamics of VNI R_{sq} of the output layer. The training is performed on the MNIST dataset 20 times, and then we evaluate the quartiles of the output VNI R_{sq} for different learning rates. Severe intensification of VNI (increases to 1) may occur as shown by the blue line which is trained with the learning rate of 10^{-2}. Otherwise VNI rises initially, and then decreases to a value which is larger than the initial VNI.

(a) Initial



(b) After 5 updates



(c) After 10 updates

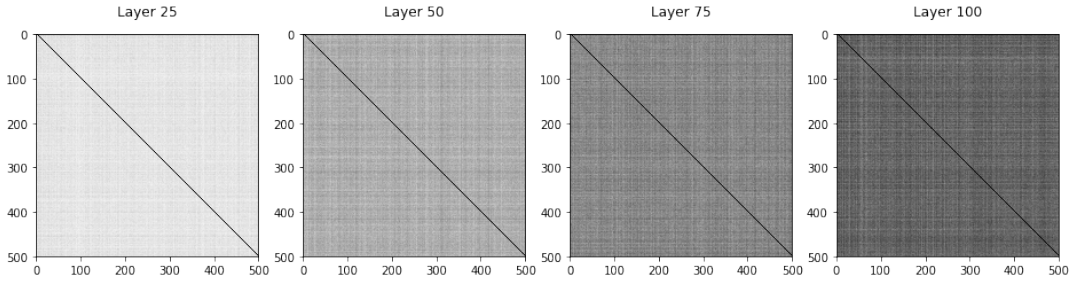
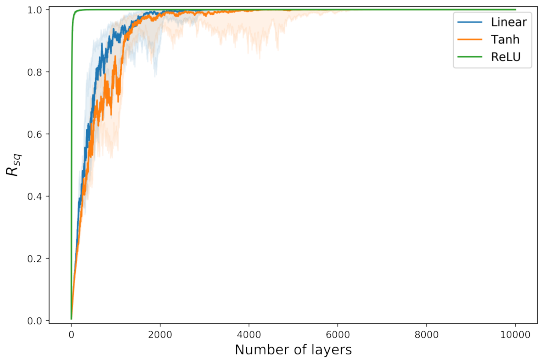


Figure 3.5: The averages of squared correlation coefficients ρ_{ij}^2 over 50 runs. It presents that overall, the correlation of each hidden layer are highly intensified.

(a) Network depth $L \in [1, 10000]$



(b) Network depth $L \in [1, 500]$

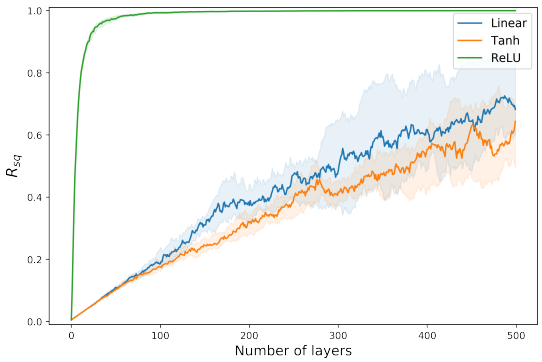
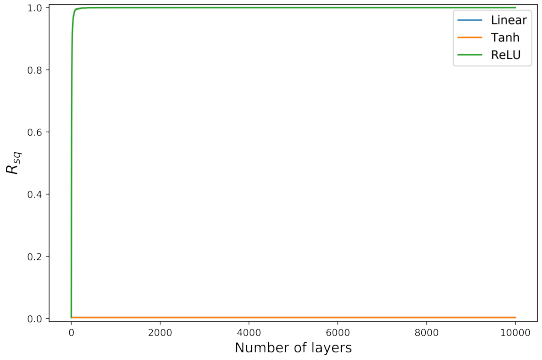


Figure 3.6

(a) Network depth $L \in [1, 10000]$



(b) Network depth $L \in [1, 500]$

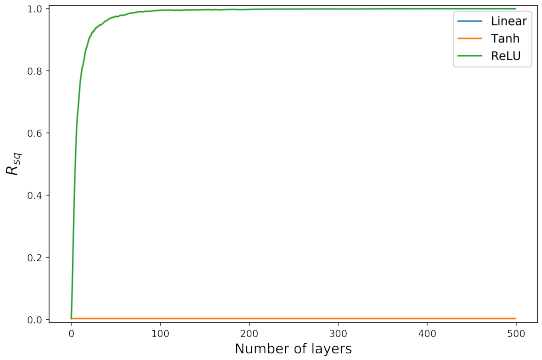


Figure 3.7

Chapter 4

Variance propagation of deep neural networks

4.1 Comparison with exploding/vanishing gradients

In this section, we explore whether the *vanishing node* phenomenon arises from the problem of exploding/vanishing gradients. Exploding/vanishing gradients in deep neural networks are a problem regarding the scale of forward-propagated signals and back-propagated gradients that exponentially explode/vanish as the networks grows deeper. We perform a theoretical analysis of exploding/vanishing gradients and show analytically the difference between them.

As in a previous study [7], we use the variances of hidden nodes to evaluate the scales of back-propagated gradients. Consider the model and the assumptions in Section 3 and an additional assumption: the gradient of output layer $\frac{\partial Cost}{\partial \mathbf{x}_L}$ is a zero-mean i.i.d. random (row) vector. That is,

$$\begin{aligned} \mathbb{E}[\mathbf{x}_0 {\mathbf{x}_0}^T] &= \sigma_x^2 \cdot \mathbf{I} \\ \mathbb{E}\Big[\Big(\frac{\partial Cost}{\partial \mathbf{x}_L}\Big)^T \frac{\partial Cost}{\partial \mathbf{x}_L}\Big] &= \sigma_y^2 \cdot \mathbf{I}, \end{aligned} \tag{4.1}$$

where σ_x^2 and σ_y^2 are defined as the variances of the input layer nodes and output layer gradients, respectively. Consider the variances of the output nodes $Var[\mathbf{x}_L]$ and input layer gradients $Var\left[\frac{\partial Cost}{\partial \mathbf{x}_0}\right]$, respectively. The exploding/vanishing gradients occur only

if the scales of forward and backward propagation exponentially increase or decrease as the depth increases. This means that the magnitude of the gradients will be bounded if we can prevent the scales of forward and backward propagation from exploding or vanishing.

According to the assumptions in Section 3 and eqn. (3.3), we can approximate the shared scalar variance of all output nodes $Var[\mathbf{x}_L] \in \mathbb{R}$ as

$$\begin{aligned} Var[\mathbf{x}_L] &= \mathbb{E}[(\mathbf{x}_L - \overline{\mathbf{x}_L})^T(\mathbf{x}_L - \overline{\mathbf{x}_L})]/N \approx \mathbb{E}[(\mathbf{J}\mathbf{x}_0)^T\mathbf{J}\mathbf{x}_0]/N \\ &= \mathbb{E}[tr(\mathbf{J}^T\mathbf{J}\mathbf{x}_0\mathbf{x}_0^T)]/N = \sigma_x^2 \cdot tr(\mathbf{J}^T\mathbf{J})/N, \end{aligned} \tag{4.2}$$

and approximation the shared scalar variance of all input gradients $Var\left[\frac{\partial Cost}{\partial \mathbf{x}_0}\right] \in \mathbb{R}$ as

$$\begin{aligned} Var\Big[\frac{\partial Cost}{\partial \mathbf{x}_0}\Big] &= \mathbb{E}\Big[\Big(\frac{\partial Cost}{\partial \mathbf{x}_0} - \overline{\frac{\partial Cost}{\partial \mathbf{x}_0}}\Big)\Big(\frac{\partial Cost}{\partial \mathbf{x}_0} - \overline{\frac{\partial Cost}{\partial \mathbf{x}_0}}\Big)^T\Big]\Big/N \\ &= \mathbb{E}\Big[\Big(\frac{\partial Cost}{\partial \mathbf{x}_L}\mathbf{J}\Big)\Big(\frac{\partial Cost}{\partial \mathbf{x}_L}\mathbf{J}\Big)^T\Big]\Big/N \\ &= \mathbb{E}\Big[tr\Big(\mathbf{J}\mathbf{J}^T\frac{\partial Cost}{\partial \mathbf{x}_L}^T\frac{\partial Cost}{\partial \mathbf{x}_L}\Big)\Big]/N \\ &= \sigma_y^2 \cdot tr(\mathbf{J}^T\mathbf{J})/N, \end{aligned} \tag{4.3}$$

where the chain rule for back-propagation: $\frac{\partial Cost}{\partial \mathbf{x}_0} = \frac{\partial Cost}{\partial \mathbf{x}_L}\frac{\partial \mathbf{x}_L}{\partial \mathbf{x}_0} = \frac{\partial Cost}{\partial \mathbf{x}_L}\mathbf{J}$ is used, and the shared scalar variance of a vector is the average of the variances of all vector components. Note that because the product of a row vector and a column vector is a scalar, the product is equal to its trace. Also, it is already known that $tr(\mathbf{J}^T\mathbf{J}) = N \cdot m_1 = N \cdot (\sigma_w^2\mu_1)^L$. Thus, we have

$$\begin{aligned} Var[\mathbf{x}_L] &\approx \sigma_x^2(\sigma_w^2\mu_1)^L \\ Var\Big[\frac{\partial Cost}{\partial \mathbf{x}_0}\Big] &= \sigma_y^2(\sigma_w^2\mu_1)^L, \end{aligned} \tag{4.4}$$

where $\sigma_w^2 = N \cdot Var[W_{ij}]$, and μ_1 is the first moment of the nonlinear activation function. It is obvious that the variances of both forward and backward propagation will neither explode nor vanish if and only if $(\sigma_w^2\mu_1) = 1$.

For the weight gradient of the hidden layer l, the variance can be used to measure the scale distribution. Because $\frac{\partial Cost}{\partial \mathbf{W}_l} = \mathbf{x}_{l-1} \cdot \frac{\partial Cost}{\partial \mathbf{h}_l}$ and both $\mathbf{x}_{l-1}$ and $\frac{\partial Cost}{\partial \mathbf{h}_l}$ are assumed to be zero-mean and independent of each other, the variance of the weight gradient can be

evaluated as

$$
\begin{aligned}
Var\Big[\frac{\partial Cost}{\partial \mathbf{W}_l}\Big] &= Var[\mathbf{x}_{l-1}] \cdot Var\Big[\frac{\partial Cost}{\partial \mathbf{h}_l}\Big] \\
&\approx \sigma_x^2(\sigma_w^2\mu_1)^{l-1} \cdot \sigma_y^2(\sigma_w^2\mu_1)^{L-l} \\
&= \sigma_x^2\sigma_y^2(\sigma_w^2\mu_1)^{L-1},
\end{aligned}
\tag{4.5}
$$

where we can evaluate $Var[\mathbf{x}_{l-1}]$ and $Var\left[\frac{\partial Cost}{\partial \mathbf{h}_l}\right]$ using the results of the forward/backward variance propagation and split the entire network into two sub-networks. One sub-network has the input layer $\mathbf{x}_0$ and output layer $\mathbf{x}_{l-1}$, and the other sub-network has the input layer $\mathbf{x}_l$ and the output layer $\mathbf{x}_L$. Note that eqn. (4.5) also concludes that if and only if $(\sigma_w^2\mu_1) = 1$, the weight gradients will never explode or vanish.

However, eqn. (3.7) shows that VNI (R_{sq}) may still accumulate with the network depth even if $(\sigma_w^2\mu_1) = 1$. That is, the characteristic of the vanishing nodes becomes evident when $(\mu_2/\mu_1^2 - 1 - s_1)$ is large, whereas vanishing/exploding gradients occurs when $(\sigma_w^2\mu_1)$ is far from 1. If the network's initialization parameter is appropriately set such that $(\sigma_w^2\mu_1)$ is close to 1, R_{sq} may still accumulate due to the network depth, the activation function, and the weight distribution. Therefore, from eqn. (3.7) and eqn. (4.5), it is clear that the problem of vanishing nodes may occur regardless of exploding/vanishing gradients.

4.2 Norm-preserving weight initialization

4.3 Batch normalization results in exploding gradients

4.4 T-normalization: A novel normalization method for deep networks

Chapter 5

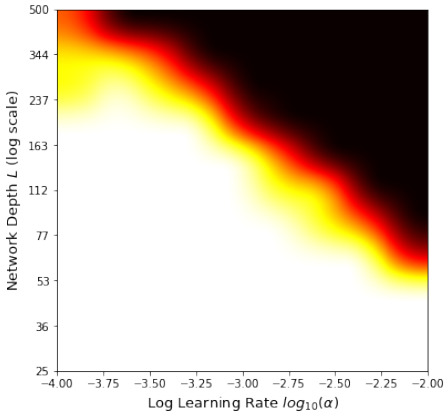
Experiments

5.1 Probability of failed training

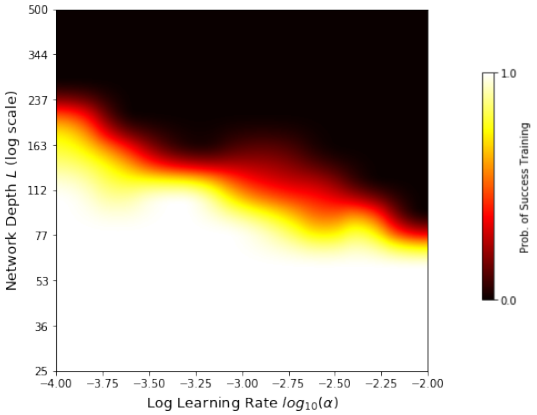
To empirically explore the effects of the phenomenon of vanishing nodes on the training of deep neural networks, we perform experiments with the training tasks on the MNIST dataset [18]. Because the purpose is to focus on the vanishing nodes, the networks are designed such that vanishing/exploding gradients will never occur; that is, they are initialized with weights ($\sigma_w^2 \mu_1 = 1$). The network is trained with 100 batch size. The number of successful training for total 20 runs is recorded to reflect the influence of vanishing nodes on the training process, which may lead to the insufficient network representation capability as shown in Figure 3.4. A successful training is considered to occur when the training accuracy exceeds 90% within 100 epochs. The network depth L ranges from 25 to 500, and the network width N is set to 500. The learning rate α ranges from 10^{-4} to 10^{-2} with the SGD algorithm. Both L and α are uniformly distributed on the logarithmic scale. The experiments are performed on the MXNet framework[3].

Figure 5.1 shows the results of two different activation functions (Tanh/ReLU) with two different weight initializations (scaled-Gaussian/orthogonal from [25]). When a network with tanh activation functions is initialized with orthogonal weights, the term of $(\mu_2/\mu_1^2 - 1 - s_1)$ in eqn. (3.7) becomes zero. Therefore, its R_{sq} will be the minimum value ($1/N$) and will not depend on the network depth. For the other network parameters, $(\mu_2/\mu_1^2 - 1 - s_1)$ will not equal zero, and R_{sq} still depends on the network depth. The ex-

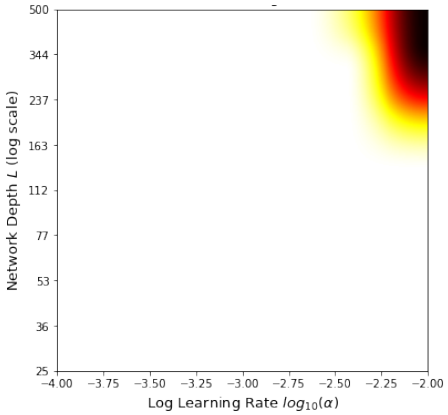
(a) Probability of Success (Tanh, Scaled Gaussian Init.)



(b) Probability of Success (ReLU, Scaled Gaussian Init.)



(c) Probability of Success (Tanh, Orthogonal Init.)



(d) Probability of Success (ReLU, Orthogonal Init.)

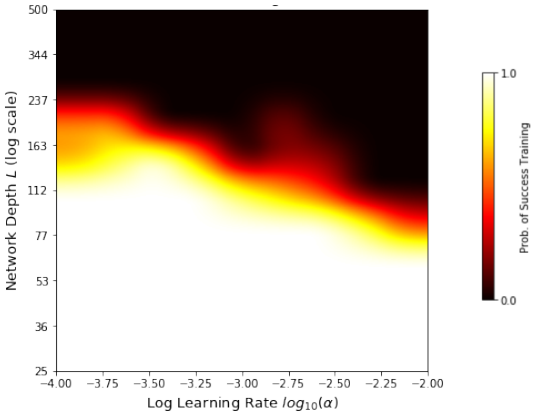
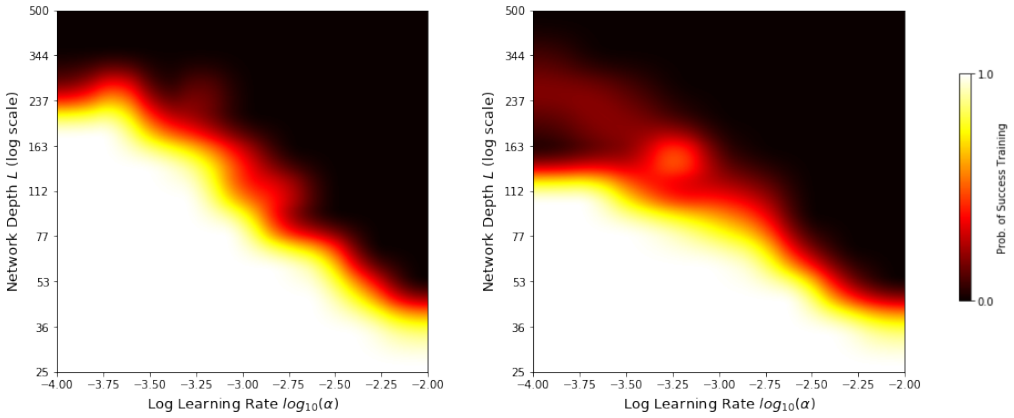


Figure 5.1: Probability of successful training for different network depth L and learning rate α (the SGD optimizer). The black color denotes zero probability of successful training.

(a) Probability of Success (Tanh, Scaled Gaussian Init.)

(b) Probability of Success (ReLU, Scaled Gaussian Init.)



(c) Probability of Success (Tanh, Orthogonal Init.)

(d) Probability of Success (ReLU, Orthogonal Init.)

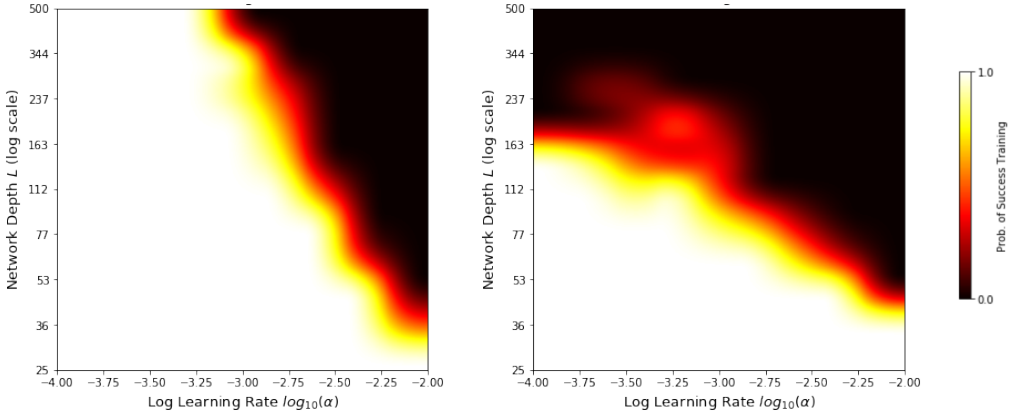


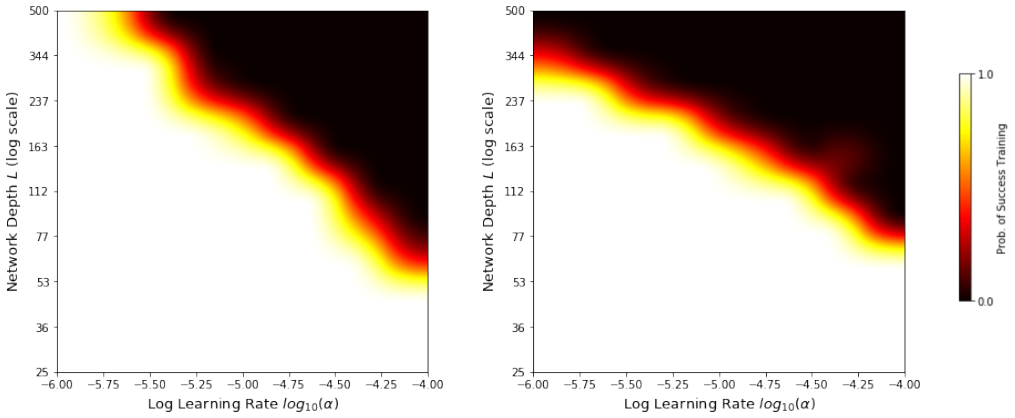
Figure 5.2: Probability of successful training for different network depth L and learning rate α (the SGD + Momentum optimizer). The networks are initialized with scaled Gaussian/orthogonal weights with Tanh/ReLu activation functions.

perimental results show the likelihood of a failed training is high when the depth L and the learning rate are large. In addition, the corresponding R_{sq} of failed cases becomes nearly 1, which causes a lack of the network representation power. It implies that the vanishing nodes problem is **the main reason** that the training fails. A comparison of Figure 5.1c with the other three results shows clearly that the networks with the minimum R_{sq} value have the highest successful training probability.

Shallow network architectures can tolerate a greater learning rate, which is why the vanishing node problem has been ignored in many networks with small depth. In a deep network, the learning rate should be set to small value to prevent R_{sq} from increasing

(a) Probability of Success (Tanh, Scaled Gaussian Init.)

(b) Probability of Success (ReLU, Scaled Gaussian Init.)



(c) Probability of Success (Tanh, Orthogonal Init.)

(d) Probability of Success (ReLU, Orthogonal Init.)

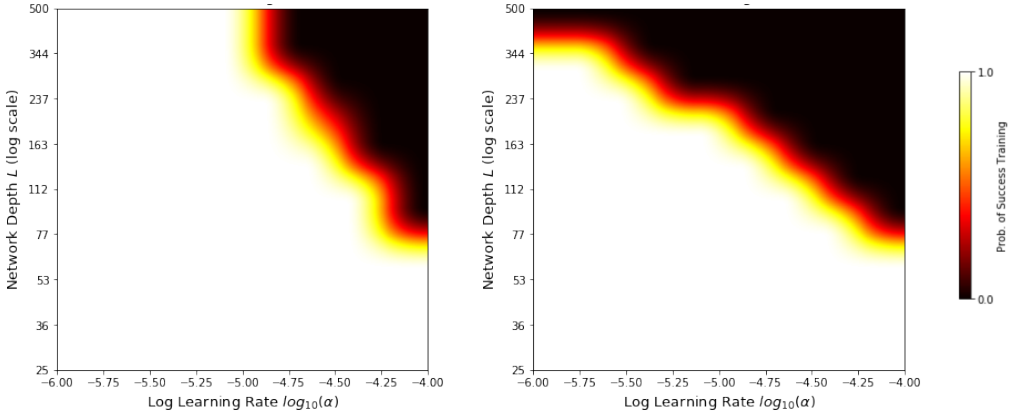


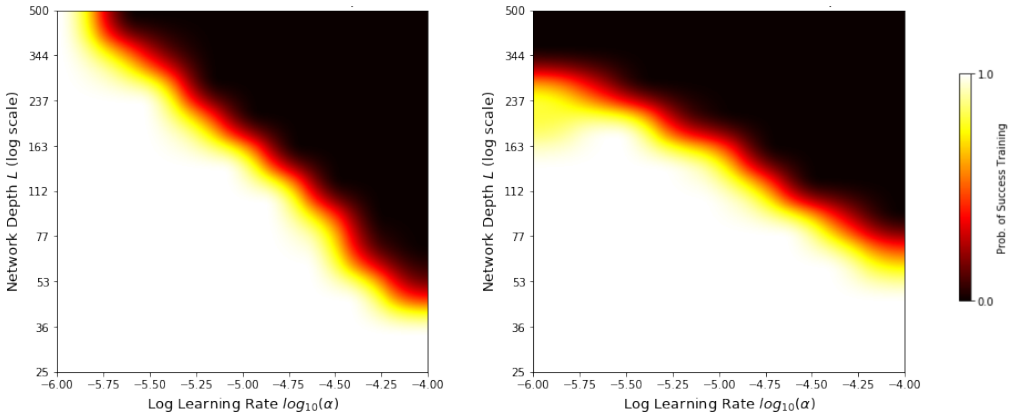
Figure 5.3: Probability of successful training for different network depth L and learning rate α (the Adam optimizer). The networks are initialized with scaled Gaussian/orthogonal weights with Tanh/ReLu activation functions.

to 1. The experimental results of various training hyperparameters (Momentum, Adam, RMSProp) are presented in Figure. 5.2, 5.3 and 5.4. Similar to the results of SGD optimization, networks with tanh activation functions and initialized with orthogonal weights have the minimum R_{sq} value, hence can achieve the highest successful training probability.

It is worth noting that, if more efficient optimization methods (e.g. Adam, RMSProp) are used, the feasible learning rate should become smaller. We can see that the boundary in Figure 5.2 has the offset to the left about 0.5 log unit comparing with Figure 5.1, and that in Figure 5.3 and 5.4 has the offset to the left about 2.0 log unit (note that the range of

(a) Probability of Success (Tanh, Scaled Gaussian Init.)

(b) Probability of Success (ReLU, Scaled Gaussian Init.)



(c) Probability of Success (Tanh, Orthogonal Init.)

(d) Probability of Success (ReLU, Orthogonal Init.)

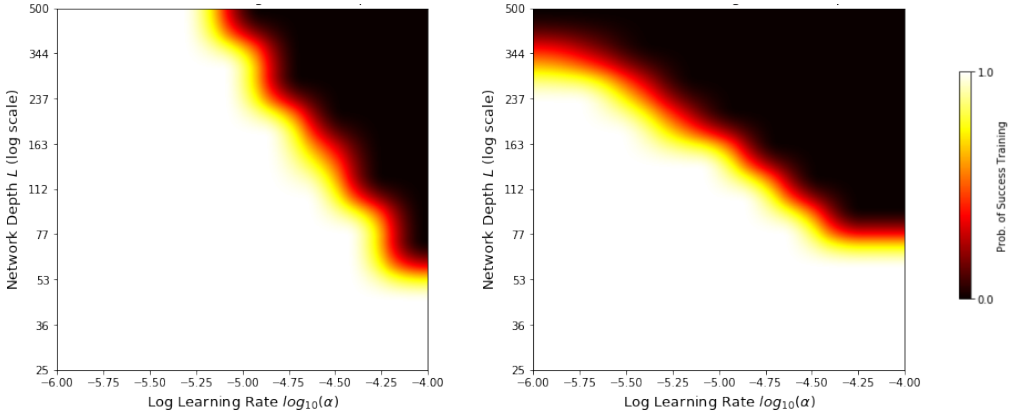


Figure 5.4: Probability of successful training for different network depth L and learning rate α (the RMSProp optimizer). The networks are initialized with scaled Gaussian/orthogonal weights with Tanh/ReLu activation functions.

the horizontal axis is 2.0 less than the range in Figure 5.1). It implies that the scale of the feasible learning rate for RMSProp and Adam should be roughly 10^2 smaller than SGD, and that for SGD+Momentum (with momentum $= 0.9$) should be about $10^{0.5}$ smaller.

The reason why the behavior of R_{sq} is effected by learning rates α remain unexplained, suggesting further investigations to better understand the relationship between learning rates and the dynamics of R_{sq} A high learning rate will cause R_{sq} to be severely intensified to nearly 1, and the representation capability of the network will be reduced, which is **the main reason** that the training fails.

5.2 Analyses of failed training

In this section, we analyze the reason why the failed training occurs from the perspectives of vanishing/exploding gradients and vanishing nodes respectively. First, the quantity σ_w^2 (the variance of weights at each layer) of trained models is collected. There are total 31,680 runs in the experiments, including 13,101 failed and 18,579 successful cases. The detailed information is presented in Table 5.1. The quantity $\sigma_w^2\mu_1$ for measuring the degree of vanishing/exploding gradients is presented in Figure 5.5 and Figure 5.6 for successful networks and failed networks. The two figures display the box and whisker plot to represent the distribution of $\sigma_w^2\mu_1$ at each network layer, and the horizontal axis indicates the depth position of the trained networks. The results show that both successful and failed networks bear the quantity $\sigma_w^2\mu_1$ near one. It indicates that both successful and failed models meet the condition of preventing networks from vanishing/exploding gradients.

Second, the difference of R_{sq} between successful and failed networks are displayed in Figure 5.7. The horizontal axis indicates the value of VNI R_{sq} of trained models evaluated by eqn. (3), and the vertical axis represents the histogram of the VNI R_{sq}. The blue histogram represents the R_{sq} of failed networks, and the orange histogram represents the R_{sq} of successful networks. The R_{sq} of failed models ranges from 0.9029 to 1.0000 with mean 0.9949 and standard deviation 0.0481, and that of successful models ranges from 0.1224 to 0.9865 with mean 0.3207 and standard deviation 0.1690. The figure shows that R_{sq} of failed networks mainly locates around 1, and that of successful networks is widely distributed. From the analysis shown in Figure 5.5, 5.6 and 5.7, it is clear that the vanishing nodes (R_{sq} reaches 1) is the main cause which makes the training failed.

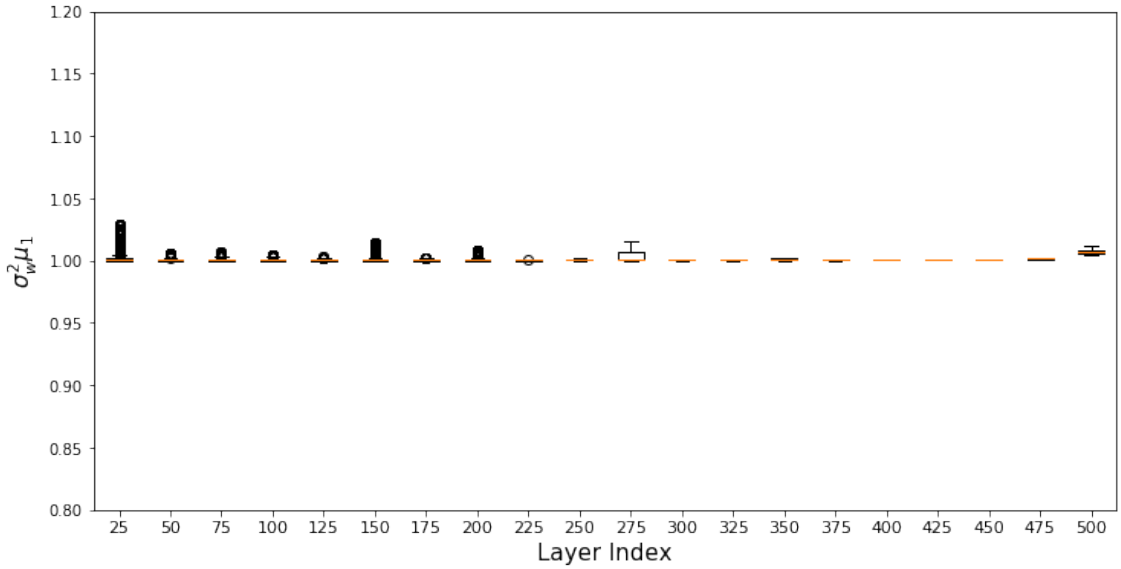


Figure 5.5: Box and whisker plot of $\sigma_w^2\mu_1$ for networks with successful training. There are 18,579 successful runs.

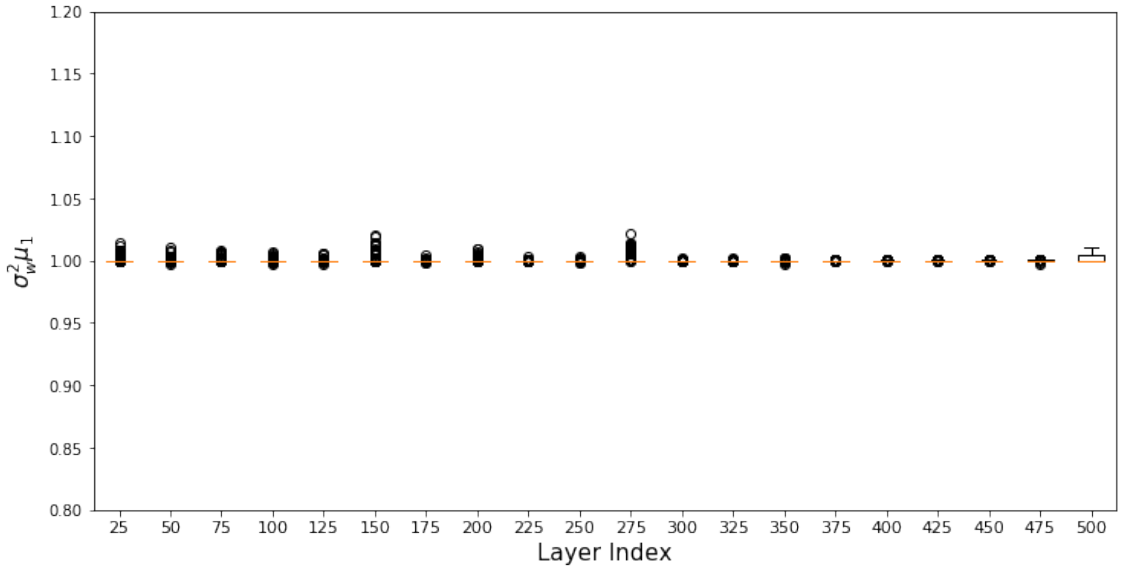


Figure 5.6: Box and whisker plot of $\sigma_w^2\mu_1$ for networks with failed training. There are 13,101 failed runs.

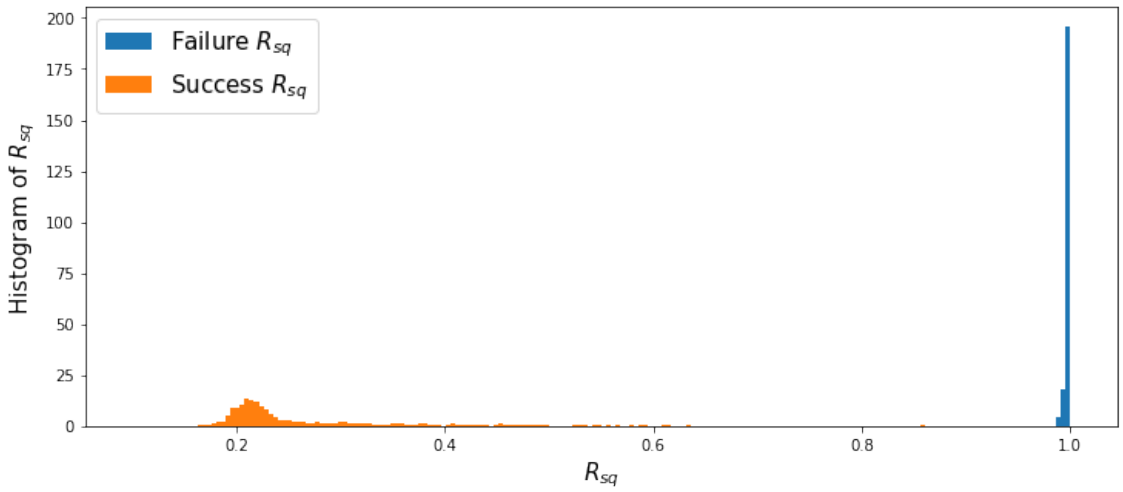


Figure 5.7: Histogram of R_{sq} of failed/successful networks.

Optimizer	Activation	Weight Init.	No. of Failure	No. of Success
SGD	Tanh	Scaled Gaussian	712	1268
	ReLU		1775	205
	Tanh	Orthogonal	62	1918
	ReLU		882	1098
SGD+momentum	Tanh	Scaled Gaussian	982	998
	ReLU		1140	840
	Tanh	Orthogonal	546	1434
	ReLU		1044	936
Adam	Tanh	Scaled Gaussian	609	1371
	ReLU		723	1257
	Tanh	Orthogonal	354	1626
	ReLU		1465	515
RMSProp	Tanh	Scaled Gaussian	763	1217
	ReLU		827	1153
	Tanh	Orthogonal	453	1527
	ReLU		764	1216

Table 5.1: The detailed numbers of successful and failed runs.

Chapter 6

Conclusion

The phenomenon of *vanishing nodes* is investigated as another challenge when training deep networks. Like the vanishing/exploding gradients problem, vanishing nodes also make training deep networks difficult. The hidden nodes in a deep neural network become more correlated as the network depth increases, so the similarity between the hidden nodes increases. Because similarity between nodes results in redundancy, the effective number of hidden nodes in a network decreases. This phenomenon is called*"vanishing nodes"*.

To measure the degree of vanishing nodes, the *Vanishing Nodes Indicator (VNI)* is proposed. It is shown theoretically that the VNI is proportional to the network depth and inversely proportional to the network width, which is consistent with the experimental results. Moreover, we explore the difference between vanishing/exploding gradients and vanishing nodes. Finally, experimental results show that vanishing/exploding gradients and vanishing nodes are two different challenges that make training deep neural networks difficult.

Bibliography

[1] R. Arora, A. Basu, P. Mianjy, and A. Mukherjee. Understanding deep neural networks with rectified linear units. *ICLR*, 2018.

[2] M. Chen, J. Pennington, and S. S. Schoenholz. Dynamical isometry and a mean field theory of rnns: Gating enables signal propagation in recurrent neural networks. *Proceedings of the 35th International Conference on Machine Learning*, 80:873–882, 2018.

[3] T. Chen, M. Li, Y. Li, M. Lin, N. Wang, M. Wang, T. Xiao, B. Xu, C. Zhang, and Z. Zhang. Mxnet: A flexible and efficient machine learning library for heterogeneous distributed systems. *CoRR*, abs/1512.01274, 2015.

[4] G. Cybenko. Approximations by superpositions of sigmoidal function. *Mathematics of Control, Signals, and Systems*, 2(4):303–314, 1989.

[5] Y. Dauphin, R. Pascanu, C. Gulcehre, K. Cho, S. Ganguli, and Y. Bengio. Identifying and attacking the saddle point problem in high-dimensional non-convex optimization. *Advances in Neural Information Processing Systems 27*, pages 2933–2941, 2014.

[6] F. Gantmacher. *The theory of matrices*. Number 1 in The Theory of Matrices. Chelsea Pub. Co., 1960.

[7] X. Glorot and Y. Bengio. Understanding the difficulty of training deep feedforward neural networks. *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, 9:249–256, 2010.

[8] I. J. Goodfellow, O. Vinyals, and A. M. Saxe. Qualitatively characterizing neural network optimization problems. *ICLR 2015*, 2014.

[9] B. Hanin. Universal function approximation by deep neural nets with bounded width and relu activations. *arXiv:1708.02691*, 2017.

[10] B. Hanin. Which neural net architectures give rise to exploding and vanishing gradients? *Neural Information Processing Systems*, 2018.

[11] B. Hanin and D. Rolnick. Complexity of linear regions in deep networks. *ICML*, 2019.

[12] K. He and J. Sun. Convolutional neural networks at constrained time cost. *CVPR*, 2015.

[13] K. He, X. Zhang, S. Ren, and J. Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. *IEEE International Conference on Computer Vision*, 2015.

[14] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[15] K. He, X. Zhang, S. Ren, and J. Sun. Identity mappings in deep residual networks. *European Conference on Computer Vision*, pages 630–645, 2016.

[16] G. Hinton, G. Dahl, A.-r. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, B. Kingsbury, and T. Sainath. Deep neural networks for acoustic modeling in speech recognition. *IEEE Signal Processing Magazine*, 29:82–97, 2012.

[17] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. *Advances in Neural Information Processing Systems*, pages 1106–1114, 2012.

[18] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[19] Y. LeCun, L. Bottou, G. B. Orr, and K.-R. Müller. Efficient backprop. *Neural Networks: Tricks of the Trade, This Book is an Outgrowth of a 1996 NIPS Workshop*, pages 9–50, 1998.

[20] J. Pennington and S. S. Schoenholz. Resurrecting the sigmoid in deep learning through dynamical isometry: theory and practice. *Advances in Neural Information Processing Systems 30*, pages 4788–4798, 2017.

[21] J. Pennington, S. S. Schoenholz, and S. Ganguli. The emergence of spectral universality in deep networks. *International Conference on Artificial Intelligence and Statistics, AISTATS 2018, 9-11 April 2018, Playa Blanca, Lanzarote, Canary Islands, Spain*, pages 1924–1932, 2018.

[22] B. Poole, S. Lahiri, M. Raghu, J. Sohl-Dickstein, and S. Ganguli. Exponential expressivity in deep neural networks through transient chaos. *Neural Information Processing Systems*, 2016.

[23] M. Raghu, B. Poole, J. Kleinberg, S. Ganguli, and J. Sohl-Dickstein. On the expressive power of deep neural networks. *arXiv:1606.05336*, 2016.

[24] D. Rolnick and M. Tegmark. The power of deeper networks for expressing natural functions. *ICLR*, 2018.

[25] A. M. Saxe, J. L. McClelland, and S. Ganguli. Exact solutions to the nonlinear dynamics of learning in deep linear neural networks. *International Conference on Learning Representations (ICLR)*, 2013.

[26] S. S. Schoenholz, J. Gilmer, S. Ganguli, and J. Sohl-Dickstein. Deep information propagation. *International Conference on Learning Representations (ICLR)*, 2017.

[27] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. van den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, S. Dieleman, D. Grewe, J. Nham, I. S. Nal Kalchbrenner, T. Lillicrap, M. Leach, K. Kavukcuoglu, T. Graepel, and D. Hassabis. Mastering the game of go without human knowledge. *Nature*, 529(7587):484–489, 2016.

[28] R. K. Srivastava, K. Greff, and J. Schmidhuber. Highway networks. *ICML 2015 Deep Learning workshop*, 2015.

[29] N. Tajbakhsh, J. Y. Shin, S. R. Gurudu, R. T. Hurst, C. B. Kendall, M. B. Gotway, and J. Liang. Convolutional neural networks for medical image analysis: Full training or fine tuning? *IEEE Transactions on Medical Imaging*, 35(5):1299–1312, 2016.

[30] M. Telgarsky. Representation benefits of deep feedforward networks. *CoRR*, abs/1509.08101, 2015.

[31] A. Veit, M. J. Wilber, and S. J. Belongie. Residual networks are exponential ensembles of relatively shallow networks. *CoRR*, abs/1605.06431, 2016.

[32] S. Wiesler and H. Ney. A convergence analysis of log-linear training. *Advances in Neural Information Processing Systems 24*, pages 657–665, 2011.

[33] Y. Wu, M. Schuster, Z. Chen, M. N. Quoc V. Le, W. Macherey, M. Krikun, Y. Cao, Q. Gao, K. Macherey, J. Klingner, A. Shah, M. Johnson, X. Liu, L. Kaiser, S. Gouws, Y. Kato, T. Kudo, H. Kazawa, K. Stevens, G. Kurian, N. Patil, W. Wang, C. Young, J. Smith, J. Riesa, A. Rudnick, O. Vinyals, G. Corrado, M. Hughes, and J. Dean. Google's neural machine translation system: Bridging the gap. *arXiv:1609.08144*, 2016.

[34] L. Xiao, Y. Bahri, J. Sohl-Dickstein, S. S. Schoenholz, and J. Pennington. Dynamical isometry and a mean field theory of cnns: How to train 10,000-layer vanilla convolutional neural networks. *Proceedings of the 35th International Conference on Machine Learning*, 2018.