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：影響深層神經網路之表現能力，並使其難以訓練的新現象
Vanishing Nodes: The Phenomena That Affects The Representation Power and The Training Difficulty of Deep Neural Networks

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

doi:10.6342/NTU201901446

國立臺灣大學（碩）博士學位論文
口試委員會審定書

國立臺灣大學
敦品勵學·愛國愛人

神經元消失：影響深層神經網路之表現能力，並使其難以訓練的新現象

Vanishing Nodes:
The Phenomena That Affects The Representation Power and The Training Difficulty of Deep Neural Networks

本論文係張文于君（r06942064）在國立臺灣大學電信工程學研究所完成之碩（博）士學位論文，於民國 108 年 07 月 10 日承下列考試委員審查通過及口試及格，特此證明

口試委員：

林宗男（簽名）
（指導教授）

林昌鴻　　李宏毅

李琳山　　吳沛遠

所　　長　[illegible]（簽名）

doi:10.6342/NTU201901446

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, 表現能力, 神經網路架構, 網路訓練問題, 正交參數初始化, 冗餘神經元, 隨機矩陣

doi:10.6342/NTU201901446

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, Representation power, Network architecture, Training difficulty, Orthogonal initialization, Node redundancy, Random matrices

doi:10.6342/NTU201901446

Contents

摘要 ii

Abstract **iii**

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 14

3.3 Relationship between the VNI and the redundancy of nodes 15

3.4 The vanishing of the representation power 18

3.5 The effect of the orthogonal weight matrices to the representation power . 21

3.6 Representation power of residual-like architectures 23

4 Variance propagation of deep neural networks **39**

4.1 Comparison of exploding/vanishing gradients and vanishing nodes 39

4.2 Norm-preserving weight initialization 41

4.3 The two obstacles for training deep nerual networks 42

5 Experiments **45**

5.1 Probability of failed training caused by vanishing nodes 45

doi:10.6342/NTU201901446

5.2 Analyses of failed training caused by vanishing nodes 50

6 Conclusion **53**

Bibliography **55**

doi:10.6342/NTU201901446

List of Figures

3.1 Scatter plots for the $linear$ activation and the scaled-Gaussian initialization. 8

3.2 Scatter plots for the $Hard\text{-}Tanh$ activation and the scaled-Gaussian initialization. 9

3.3 Scatter plots for the $ReLU$ activation and the scaled-Gaussian initialization. 10

3.4 Scatter plots for the $linear$ activation and the scaled-uniform initialization. 11

3.5 The results of VNI R_{sq} with respect to network depth L. 27

3.6 The magnitudes of correlation coefficient ρ_{ij} between output nodes. . . . 28

3.7 The dynamics of VNI R_{sq} of the output layer. 28

3.8 The dynamics of VNI R_{sq} of hidden layers. 29

3.9 The averages of squared correlation coefficients ρ_{ij}^2 over 50 runs. 30

3.10 The effective number of nodes to the VNI R_{sq} 31

3.11 Numerical simulation for eqn. (3.28) 32

3.12 The initial VNI R_{sq} of Gaussian initialized networks. 33

3.13 The initial VNI R_{sq} of orthogonal initialized networks. 34

3.14 The architectures of residual deep networks 35

3.15 The initial VNI R_{sq} of residual networks. 36

3.16 The initial VNI R_{sq} of different network architectures. 37

3.17 The initial VNI R_{sq} of convolutional neural networks. 38

4.1 The schematic diagram for deep neural network architectures 43

5.1 Probability of successful training for the SGD optimizer. 46

5.2 Probability of successful training for the SGD + Momentum optimizer. . . 47

5.3 Probability of successful training for the Adam optimizer. 48

doi:10.6342/NTU201901446

5.4 Probability of successful training for the RMSProp optimizer. 49
5.5 Box and whisker plot of $\sigma_w^2 \mu_1$ for networks with successful training. . . . 51
5.6 Box and whisker plot of $\sigma_w^2 \mu_1$ for networks with failed training. 51
5.7 Histogram of R_{sq} of failed/successful networks. 52

doi:10.6342/NTU201901446

List of Tables

3.1 The μ_k of different activation functions. 26

3.2 The s_1 of different weight distribution. 26

3.3 The maximum depth for different architectures and initializations 26

5.1 The detailed numbers of successful and failed runs. 52

doi:10.6342/NTU201901446

Chapter 1

Introduction

Deep neural networks (DNN) have succeeded in various fields, including computer vision [22], speech recognition [19], machine translation [40], medical analysis [35] and human games [33]. 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 [17, 18] 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. [9, 16] investigated this problem deeply and suggested that initializing weights in appropriate scales can prevent gradients from vanishing or exploding exponentially. [28, 32] 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

doi:10.6342/NTU201901446

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.

doi:10.6342/NTU201901446

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, [9, 16] investigated vanishing/exploding gradient propagation and gave weight initialization methods as the solution. [13] suggested that vanishing/exploding gradients might relate to the sum of the reciprocals of the hidden layer widths. [7, 11] stated that saddle points are more likely than local minima to be a problem for training deep neural networks. [15, 17, 34] 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. [31] reported dynamical isometry to theoretically ensure depth-independent learning speed. [26, 27] suggested several ways to achieve dynamical isometry for various settings of network architecture, and [4, 41] 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 [6], a single hidden layer with a finite number of neurons can approximate any continuous function on compact subsets. However, [38]

doi:10.6342/NTU201901446

states that the network depth of neural networks governs the representation power and the training performance. Theoretically, [14, 28, 29, 30] claim the expressive complexity of a network grows exponentially with its depth but not its width. For ReLU networks, [2, 12, 37] 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. [32] surveyed the propagation of the correlation between two different inputs after several layers. [24, 39] suggested that the input features must be whitened (i.e., zero-mean, unit variances and uncorrelated) to achieve a faster training speed.

doi:10.6342/NTU201901446

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

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

doi:10.6342/NTU201901446

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*.

Figure 3.1, 3.2, 3.3 and 3.4 provide intuitives view of the vanishing node problem. The network architecture is build with depth $L = 100$, width $N \in \{6, 50, 100\}$ and activation functions include $Linear$, $Hard\text{-}Tanh$ and $ReLU$. The weight matrices are initialized with scaled-Gaussian and scaled-uniform initialization (further discussion will provide in Section 4.2) and the biases are set to zeros. The network is fed with 1000 random generated data points drawn from the zero-mean white Gaussian distribution. After the forward propagation, we choose 6 output nodes from the output layer to plot scatter plots with the first output node. It is obvious that the neurons in four Subfigures (a) are so linearly correlated that the actual number of effective number of the six nodes is 1, while the neurons in four Subfigures (c) seem to be uncorrelated to each others. It implies that if the network width N is relatively small (compared to the network depth L), the output nodes may become highly correlated, and hence results in the 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

$$\frac{\partial \mathbf{x}_L}{\partial \mathbf{x}_0} = \prod_{l=1}^{L} \mathbf{D}_l \mathbf{W}_l, \tag{3.3}$$

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 [31], 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

$$\mathbf{C} \triangleq \mathbb{E}_{\mathbf{x}_0}[(\mathbf{x}_L - \overline{\mathbf{x}_L})(\mathbf{x}_L - \overline{\mathbf{x}_L})^T] \approx \mathbb{E}_{\mathbf{x}_0}[(\mathbf{J}\mathbf{x}_0)(\mathbf{J}\mathbf{x}_0)^T] = \mathbf{J}\mathbb{E}_{\mathbf{x}_0}[\mathbf{x}_0\mathbf{x}_0^T]\mathbf{J}^T = \sigma_x^2 \mathbf{J}\mathbf{J}^T, \tag{3.4}$$

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:

$$R_{sq} \triangleq \frac{\sum_{i=1}^{N} \sum_{j=1}^{N} \rho_{ij}^2 \sigma_i^2 \sigma_j^2}{\sum_{i=1}^{N} \sum_{j=1}^{N} \sigma_i^2 \sigma_j^2}. \tag{3.5}$$

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

doi:10.6342/NTU201901446

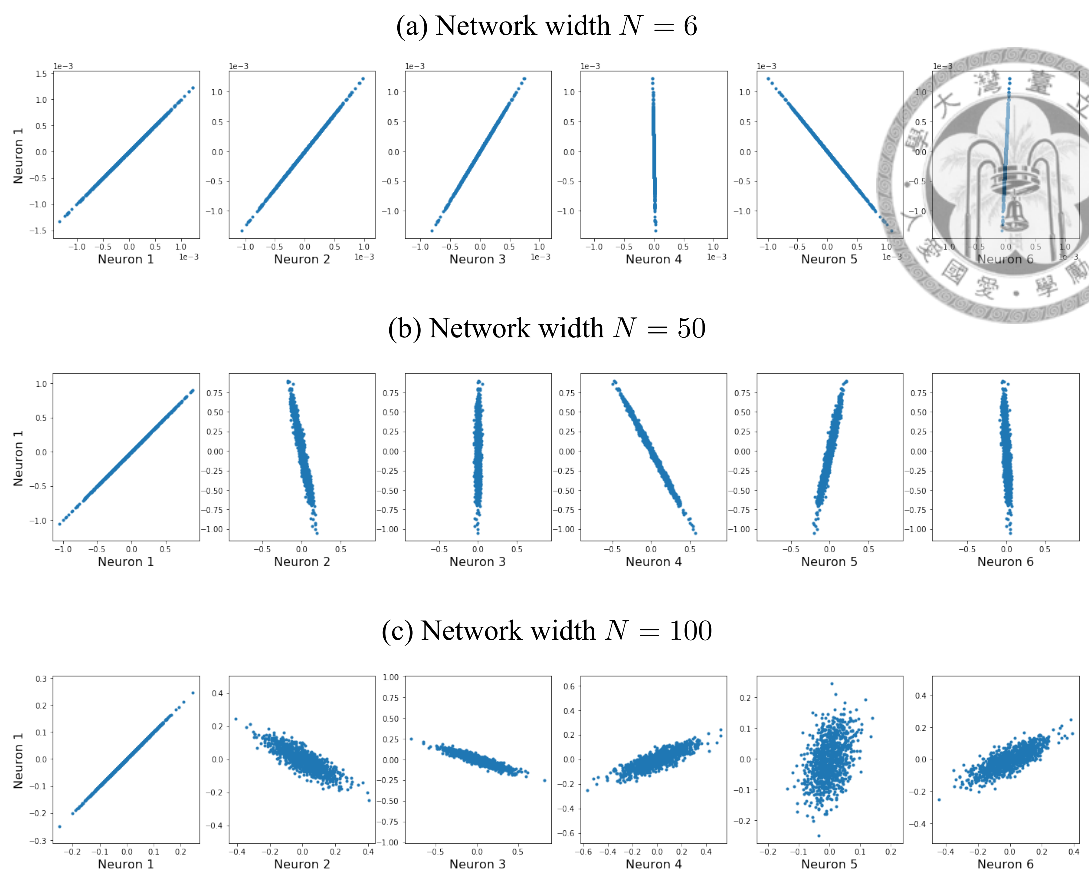


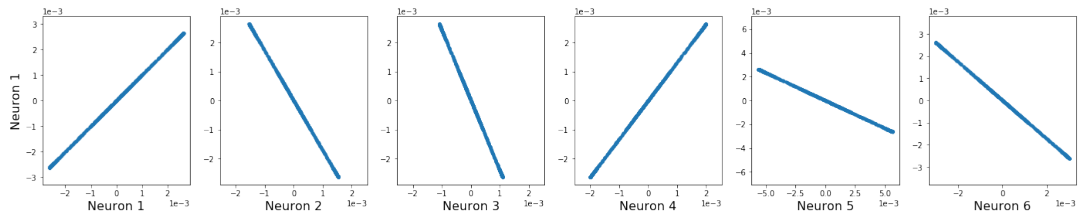
Figure 3.1: To show the output layer correlation, we plot the scatter plots of 6 random sampled output layer nodes. The network is fed with 1000 random generated data points. The network architecture is defined with network depth $L = 100$, $linear$ activation and scaled-Gaussian weight initialization.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

(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.

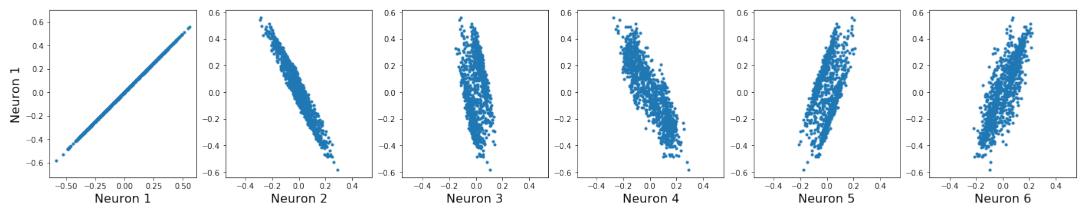
With the covariance matrix defined in eqn. (3.4) and the formulas for matrix traces,

doi:10.6342/NTU201901446

(a) Network width $N = 6$



(b) Network width $N = 50$



(c) Network width $N = 100$

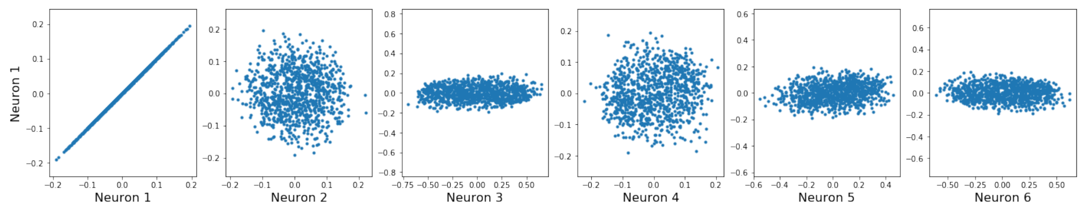
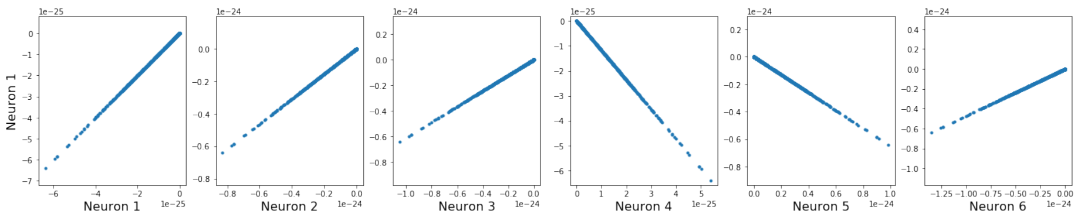


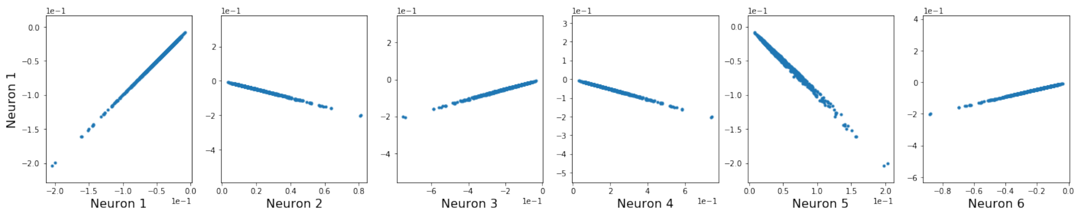
Figure 3.2: To show the output layer correlation, we plot the scatter plots of 6 random sampled output layer nodes. The network is fed with 1000 random generated data points. The network architecture is defined with network depth $L = 100$, $Hard$-$Tanh$ activation and scaled-Gaussian weight initialization. The network width $N = 6, 50, 100$ from top to bottom. We can see that correlations are much higher when the network width N is small.

doi:10.6342/NTU201901446

(a) Network width $N = 6$



(b) Network width $N = 50$



(c) Network width $N = 100$

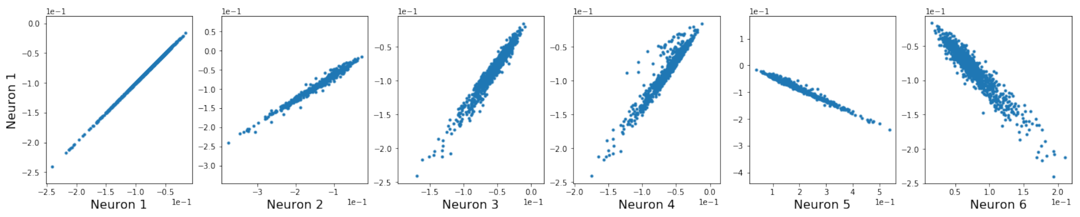
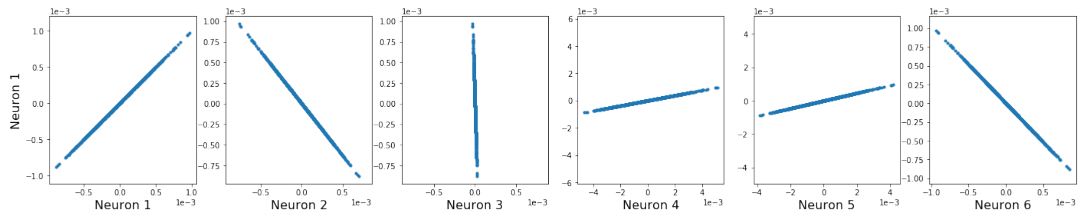


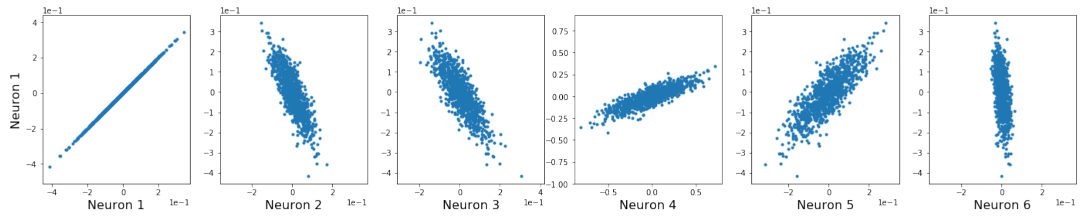
Figure 3.3: To show the output layer correlation, we plot the scatter plots of 6 random sampled output layer nodes. The network is fed with 1000 random generated data points. The network architecture is defined with network depth $L = 100$, $ReLU$ activation and scaled-Gaussian weight initialization. The network width $N = 6, 50, 100$ from top to bottom. We can see that correlations are much higher when the network width N is small.

doi:10.6342/NTU201901446

(a) Network width $N = 6$



(b) Network width $N = 50$



(c) Network width $N = 100$

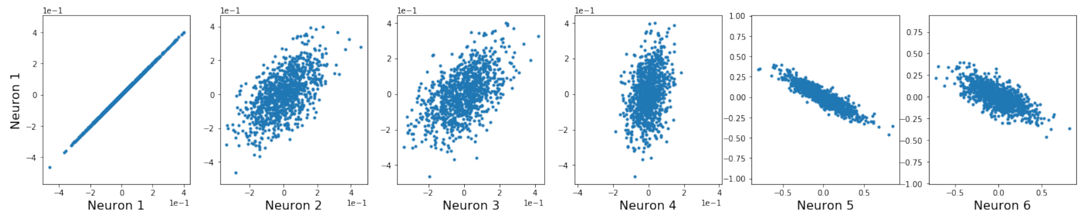


Figure 3.4: To show the output layer correlation, we plot the scatter plots of 6 random sampled output layer nodes. The network is fed with 1000 random generated data points. The network architecture is defined with network depth $L = 100$, $linear$ activation and scaled-Uniform weight initialization. The network width $N = 6, 50, 100$ from top to bottom. We can see that correlations are much higher when the network width N is small.

doi:10.6342/NTU201901446

VNI R_{sq} can be expressed as the formula of the covariance matrix as

$$\begin{aligned} R_{sq} &= \frac{\sum_{i=1}^{N}\sum_{j=1}^{N}\mathbb{E}_{\mathbf{x}_0}[(x_{L(i)} - \overline{x_{L(i)}})(x_{L(j)} - \overline{x_{L(j)}})]^2}{\sum_{i=1}^{N}\sum_{j=1}^{N}\mathbb{E}_{\mathbf{x}_0}[(x_{L(i)} - \overline{x_{L(i)}})^2]\mathbb{E}[(x_{L(j)} - \overline{x_{L(j)}})^2]} \\ &= \frac{\sum_{i=1}^{N}\sum_{j=1}^{N}[C_{(ij)}]^2}{\sum_{i=1}^{N}\sum_{j=1}^{N}C_{(ii)}C_{(jj)}} = \frac{tr(\mathbf{C}\mathbf{C}^T)}{tr(\mathbf{C})^2}, \end{aligned} \tag{3.6}$$

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

From eqn. (3.4), substituting $\sigma_x^2\mathbf{J}\mathbf{J}^T$ for **C** in eqn. (3.6), and noting that $tr(\mathbf{A}^k)$ is equal to the sum of eigenvalues to the k-th power of symmetric matrix **A** [8], an approximation of R_{sq} can be obtained:

$$R_{sq} \approx \frac{tr(\mathbf{J}\mathbf{J}^T\mathbf{J}\mathbf{J}^T)}{tr(\mathbf{J}\mathbf{J}^T)^2} = \frac{\sum_{k=1}^{N}\lambda_k^2}{(\sum_{k=1}^{N}\lambda_k)^2} = \frac{N\cdot m_2}{(N\cdot m_1)^2} = \frac{m_2}{Nm_1^2}, \tag{3.7}$$

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. (3.7), 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 [27], we can leverage the recent results by [27]: $m_1 = (\sigma_w^2\mu_1)^L$, 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. The derivation of μ_k and s_1 are given by [27], and the results are provided in Table 3.1 and 3.2. If we insert the expressions of m_1 and m_2 into eqn. (3.7), we can obtain an approximation of the expected VNI:

$$R_{sq} \approx \frac{L}{N}\Big(\frac{\mu_2}{\mu_1^2} + \frac{1}{L} - 1 - s_1\Big) = \frac{1}{N} + \frac{L}{N}\Big(\frac{\mu_2}{\mu_1^2} - 1 - s_1\Big), \tag{3.8}$$

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. (3.8) 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.

doi:10.6342/NTU201901446

To evaluate the accuracy of eqn. (3.8) with respect to the original definition in eqn. (3.5), 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 [23] are fed into the network. In each network architecture, the weights are initialized with scaled-Gaussian distribution [9] of various random seeds for 100 runs. The details of the scaled-Gaussian initialization are provided in Section 4.2. The R_{sq} calculated from eqn. (3.5) is then recorded to compute the mean and the standard deviation with respect to various network depths L. The results are shown in Figure 3.5 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. (3.8). This experiment demonstrates that VNI expressed in terms of the network parameters in eqn. (3.8) is very close to the original definition in eqn. (3.5). Similar results are obtained with different activations (e.g., Linear, ReLU) and different weight initialization (e.g., scaled uniform distribution).

Figure 3.6 plots the squared correlation coefficients between output nodes, which are evaluated with 50,000 training images in the MNIST dataset [23] for various network architectures. White indicates no correlation, and black means that $\rho_{ij}^2 = 1$. Figure 3.6 (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 3.6 (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.

In each simulation, the weight initialization follows the scaled-Gaussian distribution with the activation variance-maintaining property as defined by [9, 16]. The details of the scaled-Gaussian initialization method will be discussed in Section 4.2.

doi:10.6342/NTU201901446

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 [9], is used. The network is then trained on the MNIST dataset [23] 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 3.7.

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 3.8, the dynamics VNI R_{sq} in hidden layers is provided. The depth and the width of the network is set to $L = 100, N = 500$. The activation is $tanh$ activation and the weight matrices are initialized with scaled Gaussian distribution (will be discussed in Section 4.2). The optimization method is SGD with learning rates 10^{-3} in Figure 3.8a and 10^{-2} in Figure 3.8b. The training is performed on the MNIST dataset, and we evaluate the averages of hidden layer VNI R_{sq} over 50 runs. It shows that if we use a large learning rate like 10^{-2} for weight optimization, the VNI of over 70% of hidden layers are highly intensified in 10 updates. The dynamic of the VNI intensification starts from the output layer, and then propagated toward the input layer in an infection-like behavior. That is, the vanishing nodes problem occurs with large learning rate, and if it occurs, most of the

doi:10.6342/NTU201901446

hidden layers will be effected.

In Figure 3.9, 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 3.9a, the color of layer 100 is bright because the network width $N = 500$ is large relative to the network depth $L = 100$.

3.3 Relationship between the VNI and the redundancy of nodes

In this section, we would like to connect the VNI R_{sq} with the redundancy of nodes. First, the N random variables of node values in a hidden layer are defined as $\{X_1, X_2, ..., X_N\}$. Without loss of generality, we assume that every X_i are following $\mathcal{N}(0,1)$ distribution. Therefore, the covariance matrix of random vector $[X_1, X_2, ..., X_N]^T$ is

$$\mathbf{C} = \begin{bmatrix} 1 & \rho_{12} & \cdots & \rho_{1N} \\ \rho_{21} & 1 & \cdots & \rho_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ \rho_{N1} & \rho_{N2} & \cdots & 1 \end{bmatrix}, \tag{3.9}$$

where $C_{ij} = \mathbb{E}[(X_i - \overline{X_i})(X_j - \overline{X_j})]$ as defined in eqn. (3.4), ρ_{ij} is the correlation coefficient between X_i and X_j, and hence $\mathbf{C}$ is a symmetric matrix. By the definition of the VNI R_{sq} from eqn. (3.5), the R_{sq} can be represented as

$$R_{sq} = \frac{\sum_{i=1}^{N}\sum_{j=1}^{N}\rho_{ij}^2}{N^2} = \frac{tr(\mathbf{C}\mathbf{C}^T)}{N^2} = \frac{1}{N^2}tr(\mathbf{C}^2). \tag{3.10}$$

Let the eigenvalues of $\mathbf{C}$ are $\lambda_1, \lambda_2, \ldots, \lambda_N$. By the relationship between matrix trace

doi:10.6342/NTU201901446

and eigenvalues, we have

$$\begin{aligned}\sum_{i=1}^{N}\lambda_i &= tr(\mathbf{C}) = N \\ \sum_{i=1}^{N}\lambda_i^2 &= tr(\mathbf{C}^2) = N^2 R_{sq}.\end{aligned} \tag{3.11}$$

To relate R_{sq} with the redundancy of random variables, a method for measuring the redundancy is needed. From the principle component analysis (PCA), the eigenvalue of **C** can represent the energy (i.e. the variance) associated with each eigenvector. Therefore, we can use the distribution of eigenvalues λ_i to determine the proportion of redundant components, and hence the effective number of nodes can be evalueated. Similar to PCA, we first rearange the order of eigenvalues λ_i such that $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_N \geq 0$. We define a constant $\varepsilon \in (0,1)$ to be the "effective threshold ratio" of eigenvalues. That is, if $\lambda_i \geq \varepsilon\lambda_1$, then we say that the i-th component λ_i is ε-effective. Otherwise, the i-th component λ_i is said to be redundant.

We introduce a new metric called "ε-effective number of nodes (ε-ENN)" as

$$\varepsilon\text{-ENN} \equiv N_e^\varepsilon \triangleq max(\{t \in \mathbb{N} : \lambda_t \geq \varepsilon\lambda_1\}). \tag{3.12}$$

That is, ε-ENN is the maximum number of ε-effective nodes. As in eqn. (3.11), the constraints on λ_i are already derived. Also, it is intuitive that the maximum in eqn. (3.12) can simply be attained with eigenvalues $\{\lambda_1, \varepsilon\lambda_1, \ldots, \varepsilon\lambda_1, 0, \ldots, 0 \ldots, 0\}$, where there are $(N_e^\varepsilon - 1)$ $\varepsilon\lambda_1$ and $(N - N_e^\varepsilon)$ zeros. Insert these eigenvalues into eqn. (3.11), we have

$$\begin{aligned}\lambda_1 + (N_e^\varepsilon - 1)\varepsilon\lambda_1 &= N \\ \lambda_1^2 + (N_e^\varepsilon - 1)(\varepsilon\lambda_1)^2 &= N^2 R_{sq}.\end{aligned} \tag{3.13}$$

Inserting the first equation in eqn. (3.13) into the second one, we can get

$$[1 + (N_e^\varepsilon - 1)\varepsilon]^2 = \left(\frac{N}{\lambda_1}\right) = \frac{1 + (N_e^\varepsilon - 1)\varepsilon^2}{R_{sq}}, \tag{3.14}$$

doi:10.6342/NTU201901446

which is a solvable quadratic equation. The numerical solution of the effective number of nodes for given ε are provided in Figure 3.10.

Theorem 1 (Opposite trend between the VNI and the ENN)**.** *For any threshold ε, the ε-effective number of nodes strictly decreases as the VNI R_{sq} increases.*

Proof. Evaluate the differentiation of both sides of the eqn. (3.14), the derivative of R_{sq} with respect to N_e^ε is

$$\begin{aligned}\frac{dR_{sq}}{dN_e^\varepsilon} &= \frac{[1+(N_e^\varepsilon-1)\varepsilon]^2\cdot N_e^\varepsilon\varepsilon^2-[1+(N_e^\varepsilon-1)\varepsilon^2]\cdot N_e^\varepsilon\cdot 2\varepsilon[1+(N_e^\varepsilon-1)\varepsilon]}{[1+(N_e^\varepsilon-1)\varepsilon]^4}\\ &=\varepsilon\cdot\frac{[1+(N_e^\varepsilon-1)\varepsilon]\cdot N_e^\varepsilon\varepsilon-2[1+(N_e^\varepsilon-1)\varepsilon^2]}{[1+(N_e^\varepsilon-1)\varepsilon]^3}\\ &=\varepsilon\cdot\frac{\varepsilon-2-(N_e^\varepsilon-1)\varepsilon^2}{[1+(N_e^\varepsilon-1)\varepsilon]^3}\\ &<0.\end{aligned}\tag{3.15}$$

Since in eqn. (3.15), the derivative of R_{sq} with respect to N_e^ε is always negative, the N_e^ε is strictly decreasing with R_{sq}.

□

Theorem 2 (Network collapsing)**.** *For any threshold ε,
the ε-effective number of nodes N_e^ε becomes only 1 when the VNI R_{sq} is 1.*

Proof. Insert the $R_{sq}=1$ condition into eqn. (3.14), then we have

$$\begin{aligned}&[1+(N_e^\varepsilon-1)\varepsilon]^2=1+(N_e^\varepsilon-1)\varepsilon^2\\ \Rightarrow&(N_e^\varepsilon-1)[(N_e^\varepsilon-2)\varepsilon+2]=0\\ \Rightarrow&N_e^\varepsilon=1\end{aligned}\tag{3.16}$$

The solution of N_e^ε in eqn. (3.14) is 1 (no matter the value of ε). Therefore, the ε-effective number of nodes N_e^ε becomes only 1 when the VNI R_{sq} is 1. □

By Theorem 1, and Theorem 2, we can say that the effective number of nodes in a layer of a network vanishes to 1 as the VNI R_{sq} increases to 1. If the output layer of a network has the VNI $R_{sq}=1$, then we can say that the network suffers from the "*network*

doi:10.6342/NTU201901446

collapsing" problem, which has only 1 effective node at the output layer and hence cannot solve most of training tasks.

3.4 The vanishing of the representation power

The phenomena that the representation power of a very deep network vanishes is shown in this section. Recent works [2, 12, 14, 28, 29, 30, 37] put emphasis on the benifit of increasing the depth of a neural network, and claim that the representation power of a neural network grows exponentially as its depth. However, the representation power discussed in previous results is mainly the theoretical upper bound of all variable space. Practically, it has small probability for the representation power to reach the upper bound when the weight matrices are randomly initialized. In the following, we will show that if the weight matrices are drawn from a non-orthogonal probability distribution (such as the normal distribution and the uniform distribution), the VNI R_{sq} increases to 1 as the network goes deeper. Moreover, as the VNI R_{sq} reaches nearly 1, the representation power of the network will vanish.

The VNI R_{sq} has been defined in eqn. (3.5) to measure the correlation between the output nodes of a network, and hence the VNI can be viewed as a measurement of the level of redundancy. In Section 3.3, the redundant number of nodes has already been connected to the VNI R_{sq}. Therefore, we can simply use the VNI R_{sq} as an approximation of the ratio of redunt nodes, and the number of effective nodes can be approximated as $N \cdot (1 - R_{sq}) +$ 1, that is, one node along with other non-redundant nodes. The representation power is closely related to the effective number of network nodes, and the VNI R_{sq} can provide an estimation of the effective number of nodes (as in Section 3.3).

In Figure 3.12, the network width N is set to 500 and the network depth L ranges from 1 to 10000. The network is fed with 1000 randomly generated data point drawn from zero-mean white Gaussian distribution with standard deviation equals to 0.1. The VNI R_{sq} are evaluated according to the definition in eqn. (3.5). The weights are initialized with scaled-Gaussian distribution ([9, 16]), which will be discussed in Section 4.2, and the biases are initialized to zeros. The activation functions of the networks include tanh,

doi:10.6342/NTU201901446

ReLU and linear. Note that for the ReLU case, we add the layer normalization ([3]) blocks between hidden layers in order to prevent the node values from converging to zeros. If the node values converges to zeros, the result of eqn. (3.5) become undefined. Also note that since in our case, the layer normalization only rescales the node values, it will not affect the value of VNI evaluated from eqn. (3.5), which is a scale-invariant metric. The simulation is repeated for 20 times, and the medians of the VNI over 20 runs are plotted as the solid lines, and the boundaries of the colored regions are the first and the third quartiles of the VNI. It is shown that for a feed-forward architecture under a non-orthogonal initialization, the initial VNI R_{sq} increases to 1 as L gets larger. That is, the representation power vanishes as the network goes deeper. The VNI of ReLU activation, especially, grows in the steepest with the network depth, and the reason can be observed from the eqn. (3.8), Table 3.1 and Table 3.2. The R_{sq} approximated by eqn. (3.8) for ReLU activation is $(2L+1)/N$ while the R_{sq} for linear and tanh activation is nearly $(L+1)/N$. Therefore, the ReLU activation suffers more from the vanishing representation power.

We define the ”maximal depth” as the maximal L such that in less than half of 20 runs(i.e. 10 runs), the approximated number of effective nodes achieve greater than 1. The maximal depths for different activation functions, weight initializations and network architectures are shown in Table 3.3. It shows that the maximal depth of ReLU activation is much less than linear and tanh activations. It has been stated [10] that the ReLU activation provides a sparse and distributed representation That is, the ReLU activation selects half of neurons as the active nodes, which may reduce the effective number of nodes. Since the maximal depth is closely related to the effective number of nodes, it provides another aspect explaining that the representation power vansihes faster for the ReLU activation.

Here, we provide a theoretical claim for reasoning the VNI R_{sq} of a Gaussian initialized linear network.

Theorem 3 (The VNI R_{sq} goes to 1 as $L \to \infty$)**.** *For a linear network with Gaussian initialized weight matrices $\mathbf{W}_l, l \in \{1, 2, \ldots, L\}$, the VNI R_{sq} of the network goes to 1 as $L \to \infty$.*

Proof. Let the product of weight matrices $\mathbf{P}$ as the input-output Jacobian (defined in

doi:10.6342/NTU201901446

eqn. (3.3)) of the linear network

$$\mathbf{P}_L \equiv \prod_{l=1}^{L} \mathbf{W}_l. \tag{3.17}$$

From eqn. (3.7), we would like to show that $R_{sq} = \frac{\sum_{k=1}^{N} \lambda_k^2}{(\sum_{k=1}^{N} \lambda_k)^2}$ goes to 1 as $L \to \infty$. Note that the λ_k is the k-th eigenvalue of $\mathbf{P}_L\mathbf{P}_L^T$, satisfying $\lambda_1 \leq \lambda_2 \leq \cdots \leq \lambda_N$.

Considering the asymptotic behavior of eigenvalues λ_k when the depth L tends to infinity, we can apply the Lyapunov exponents to the matrix $\mathbf{P}_L\mathbf{P}_L^T$ as in [21].

$$\lim_{L\to\infty} (\mathbf{P}_L\mathbf{P}_L^T)^{1/2L} = e^{\mathbf{H}}, \tag{3.18}$$

where **H** has eigenvalues $\mu_1 \leq \mu_2 \leq \cdots \leq \mu_N$, which are known as the Lyapunov exponents.

By the Oseledec's theorem [25], we have the asymptotic value of eigenvalue

$$\lambda_k \sim e^{2L\mu_k}. \tag{3.19}$$

Previous works ([21]) have already derived the value of μ_k

$$\mu_k = \frac{1}{2}\log 2 + \frac{1}{2}\psi\Big(\frac{k}{2}\Big), \tag{3.20}$$

where $\psi(\cdot)$ denotes the *digamma function*, which is defined as the logarithmic derivative of the gamma function. As in [1], the digamma function has the asymptotic expansion

$$\psi(x) \sim \log x - \frac{1}{2x} + O(x^{-2}). \tag{3.21}$$

Therefore, the eigenvalues λ_k has the asymptotic approximation by eqn. (3.19) to eqn. (3.21)

$$\lambda_k \sim e^{L(\log k)} = k^L. \tag{3.22}$$

doi:10.6342/NTU201901446

Insert the result of eqn. (3.22) into eqn. (3.7), we have

$$\begin{aligned}
\lim_{L\to\infty} \frac{\sum_{k=1}^{N} \lambda_k^2}{\left(\sum_{k=1}^{N} \lambda_k\right)^2} &= \lim_{L\to\infty} \frac{\sum_{k=1}^{N} k^{2L}}{\left(\sum_{k=1}^{N} k^L\right)^2} \\
&= \lim_{L\to\infty} \frac{\sum_{k=1}^{N} (k/N)^{2L}}{\left(\sum_{k=1}^{N} (k/N)^L\right)^2} \\
&= 1.
\end{aligned} \tag{3.23}$$

That is, the VNI R_{sq} of a linear network with Gaussian initialized weight matrices goes to 1 as $L \to \infty$.

$\square$

Also, for a nonlinear network with norm-preserving Gaussian initialized weight matrices (see Section 4.2 for further discussion), we can perform a linear approximiation on the input-output Jacobian matrix. That is, replace the product matrix defined in eqn. (3.17) with the Jacobian matrix defined in eqn. (3.3). Because the weight matrices are norm-preserving, then we have the same distribution of $\mathbf{D}_l$ for all hidden layers. That is,

$$\mathbb{E}[\mathbf{D}_l] = diag(\mathbb{E}[\phi'(x_{l(1)})], \mathbb{E}[\phi'(x_{l(2)})], \ldots, \mathbb{E}[\phi'(x_{l(N)})]) \approx \psi \mathbf{I} \tag{3.24}$$

where the constant ψ denote the expected derivative $\mathbb{E}[\phi'(x_{l(i)})]$ for every node i. After the linear approximation in eqn. (3.24), we can apply the similar procedure in the proof of Theorem 3, and then we can obtain the result for non-linear networks. That is, the VNI R_{sq} of a non-linear network with norm-preserving Gaussian initialized weight matrices goes to 1 as $L \to \infty$.

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

The orthogonal weight initialization [31] is one of methods that train a very deep neural network successfully. In [41], the trainable depth of a convolutional neural network with orthogonal initialized weights can reach over 10000 layers. In this section, we will show

doi:10.6342/NTU201901446

that the orthogonal weight matrices can prevent the representation power of networks from vanishing.

In Figure 3.13, similar settings to Figure 3.12 are used excluding the weights, which are initialized with scaled-Orthogonal distribution ([31], further discussion is provided in Section 4.2). It is shown that for a feed-forward architecture under the orthogonal initialization with linear or tanh activation function, the initial VNI R_{sq} remain nearly minimum (1/N) even when the network depth L gets larger, and that the VNI of ReLU activation increases to 1 as the network goes deeper.

From eqn. (3.8) and Table 3.2, it can be derived that the approximation of the VNI R_{sq} of networks with orthogonal weight matrices is $\left[\left(\frac{\mu_2}{\mu_1^2} - 1\right)L + 1\right] \Big/ N$. That is, if an activation function such that $\frac{\mu_2}{\mu_1^2} \sim 1$ is chosen, then the VNI R_{sq} can remain in a lower level even when the network goes much deeper. Therefore, it explains that the network with linear and tanh activation has a constant VNI R_{sq} with respect to the network depth, and the VNI R_{sq} for the network with ReLU activation still increases to 1.

To provide a more intuitive explanation, we first consider the linear activation case. Because the product of orthogonal matrices remain an orthogonal matrix, the input-output correlation for a 10000-layer network with orthogonal weights can be reduced to a 1-layer shallow network with a single orthogonal weight. Therefore for such a network setting, the representation power remain the same even if the network depth L goes arbitrarily large.

Consider the ReLU activation network with orthogonal weight matrices. The output layer is $\mathbf{x}_L = \phi(\mathbf{W}_L \ldots \phi(\mathbf{W}_2 \phi(\mathbf{W}_1 \mathbf{x}_0)))$. Analytically, the ReLU activation function $\phi(\cdot)$ can be taken as a matrix $\mathbf{U}_l$ with all the off-diagonal elements equal to zeros and the diagonal elements $\in \{0, 1\}$, which are dependent to the input data. That is, the output layer can be expressed as $\mathbf{x}_L = (\mathbf{U}_L \mathbf{W}_L) \ldots (\mathbf{U}_2 \mathbf{W}_2)(\mathbf{U}_1 \mathbf{W}_1)\mathbf{x}_0$. Note that the matrices $(\mathbf{U}_l \mathbf{W}_l)$ can be viewed as the random samples from the row vectors of the orthogonal matrix $\mathbf{W}_l$. Since the random sample operation is equivalent to the random projection onto the coordinate hyperplanes, the orthogonal property of $\mathbf{W}_l$ will be no longer kept after the multiplication with $\mathbf{U}_l$. Therefore, the representation power of the network faced

doi:10.6342/NTU201901446

the same problem as stated in Section 3.4.

The maximal depths for different activation functions with orthogonal weight initializations are also shown in Table 3.3. It shows that the maximal depths of linear and tanh activations are more than 10000 layers, but that of ReLU activation is much small, which is consistent to our analysis.

3.6 Representation power of residual-like architectures

Residual network ([17, 18]) is another example for training very deep nerual networks successfully. Unlike the orthogonal weight initialization, the residual network is one of an architecture solution to improve the training performance. In this section, the advantage of the additional identity skip connections (i.e. residual shortcut connections) and the effect to the representation power will be discussed.

First, the network architecture to be considered is present in Figure 3.14, which is similar to [18]. For the 1-layer shortcut case in Figure 3.14a, we can rewrite the network equation defined in eqn. (3.1) as

$$\begin{aligned} \mathbf{x}_l &= \mathbf{x}_{l-1} + \mathcal{F}(\mathbf{W}_l, \mathbf{x}_{l-1}) \\ &= \mathbf{x}_0 + \sum_{i=0}^{l-1} \mathcal{F}(\mathbf{W}_{i+1}, \mathbf{x}_i), \end{aligned} \tag{3.25}$$

where $\mathcal{F}(\cdot)$ is the residual block function consist of batch normalization ([20]), activation function and weight multiplication. After the first-order approximation as eqn. (3.4), the residual block function $\mathcal{F}(\mathbf{W}_l, \mathbf{x}_{l-1})$ can be approximated as $\mathbf{F}_l\mathbf{x}_{l-1}$. Therefore, the input-output Jacobian matrix $\mathbf{J}$ can be expressed as $\prod_{l=1}^{L}(\mathbf{I} + \mathbf{F}_l)$, where $\mathbf{I}$ is the identity matrix. The whole network hence can be viewed as a feed-forward network with weight matrices $\widehat{\mathbf{F}}_l \equiv (\mathbf{I} + \mathbf{F}_l)$.

We would like to show that the increasing effect to the VNI R_{sq} of $\widehat{\mathbf{F}}_l$ is much less than that of $\mathbf{F}_l$ by comparing the expected cosine similarity between column vectors of $\widehat{\mathbf{F}}_l$ and $\mathbf{F}_l$. Since the hidden nodes of $\mathbf{x}_l$ can be approximated as a result of linear transformation of $\mathbf{x}_{l-1}$ with column vectors of $\widehat{\mathbf{F}}_l$ or $\mathbf{F}_l$, the VNI R_{sq} of layer l, which is the weighted

doi:10.6342/NTU201901446

average of correlation coefficient of $\mathbf{x}_l$, is highly dependent on the magnitude of the cosine similarity between column vectors of the transformation matrix.

First, the cosine similarity between two vectors is defined as

$$cos(\mathbf{a}, \mathbf{b}) \triangleq \frac{\mathbf{a}^T\mathbf{b}}{||\mathbf{a}|| \cdot ||\mathbf{b}||}, \tag{3.26}$$

where $||\mathbf{v}||$ denotes the L_2-norm of vector $\mathbf{v}$ (i.e. $\sqrt{\mathbf{v}^T\mathbf{v}}$). Note that the transformation matrix $\mathbf{F}_l$ is basically the product of $\mathbf{D}_l$ (defined in eqn. (3.3)) and $\mathbf{W}_l$, and therefore the $\mathbf{F}_l$ inherits the zero-mean property of $\mathbf{W}_l$. Let the i-th column vectors of $\widehat{\mathbf{F}_l}$ and $\mathbf{F}_l$ be written as $\widehat{\mathbf{f}}_{l(i)}$ and $\mathbf{f}_{l(i)}$. By the fact that $\widehat{\mathbf{F}_l} \equiv (\mathbf{I} + \mathbf{F}_l)$, we have

$$\begin{aligned} \widehat{\mathbf{f}}_{l(i)} &= \mathbf{f}_{l(i)} + [\underbrace{0, \ldots, 0}_{(i-1)}, 1, \underbrace{0, \ldots, 0}_{(N-i)}]^T \\ &= [f_{l(i,1)}, \ldots, f_{l(i,i-1)}, (f_{l(i,i)} + 1), f_{l(i,i+1)}, \ldots, f_{l(i,N)}]^T. \end{aligned} \tag{3.27}$$

Therefore, the cosine similarity between i-th and j-th column vectors of $\widehat{\mathbf{F}_l}$ can be expressed as

$$\begin{aligned} cos(\widehat{\mathbf{f}}_{l(i)}, \widehat{\mathbf{f}}_{l(j)}) &\equiv \frac{\widehat{\mathbf{f}}_{l(i)}^T \widehat{\mathbf{f}}_{l(j)}}{||\widehat{\mathbf{f}}_{l(i)}|| \cdot ||\widehat{\mathbf{f}}_{l(j)}||} \\ &= \frac{\mathbf{f}_{l(i)}^T \mathbf{f}_{l(j)} + f_{l(i,j)} + f_{l(j,i)}}{\sqrt{||\mathbf{f}_{l(i)}||^2 + 2f_{l(i,i)} + 1} \cdot \sqrt{||\mathbf{f}_{l(j)}||^2 + 2f_{l(j,j)} + 1}}, \end{aligned} \tag{3.28}$$

which has a smaller magnitude (compared with $cos(\mathbf{f}_{l(i)}, \mathbf{f}_{l(j)})$) with high probability (as shown in Figure 3.11). That is, the residual-like transformation is closer to the orthogonality than the vanilla feed-forward transformation.

That is, compared with the original feed-forward network with the layer-by-layer transformation $\mathbf{F}_l$, the residual-like architecture with identity shortcut connection has the layer-by-layer transformation $\widehat{\mathbf{F}_l} = \mathbf{I} + \mathbf{F}_l$, which is more close to the orthogonality even when the activation funciton is ReLU. From the results of Section 3.5, we can say that the residual-like architecture suffer less from the vanishing representation power problem.

In Figure 3.15, similar settings to Figure 3.12 are used excluding the network architectures, which are defined in Figure 3.14. It is shown that for a residual-like architecture, the

doi:10.6342/NTU201901446

VNI R_{sq} grows slowly as the network depth L gets larger. The ReLU activation function, which results in the vanishing representation power in Section 3.4 and 3.5, does not make the VNI R_{sq} go to 1 in 10000 layers. Instead, the VNI R_{sq} for a 10000-layer residual ReLU network grows to a relatively small value (~ 0.04), which implies that the network maintains enough representation power even when the network is very deep. Also, the VNI R_{sq} for the 2-layer skip grows more slowly than that for the 1-layer skip. It can be reasoned that the 2-layer skip architecture, in some sense, reduces the effective network depth to nearly $L/2$ because the input-output Jacobian for the 2-layer skip architecture can be expressed as $\prod_{i=1}^{L/2}(\mathbf{I} + \mathbf{F}_{2i-1}\mathbf{F}_{2i})$.

The maximal depths of residual-like network for different activation functions with scaled-Gaussian weight initialization are also shown in Table 3.3. It shows that no matter which activation function is chosen, the maximal depths of the networks are more than 10000 layers. That is, the residual-like architecture can prevent the network representation power from intensely vanishing. Similar results can be observed in Figure 3.16.

Also, in Figure 3.17, we perform the simulation on convolutional neural networks. The feature sizes of hidden layers are all the same (32 width, 32 height and 5 channels), and the network is fed with 1000 randomly generated image following the zero-mean white Gaussian. The stride and the dilation are set to 1, and pooling layers are not inserted into the network. The VNI R_{sq} is evaluated with flatten vectors. The median, the first and third quartiles of VNI R_{sq} over 20 runs are presented. It shows that the convolution operation can make the VNI R_{sq} increase slower with respect to the network depth L, and similar to Figure 3.16, the residual connection helps the network maintain the representation power. Also, if $tanh$ activation is chosen, then the orthogonal weight initialization can also keep the VNI R_{sq} at the minimal value, and thus maintain the representation power.

doi:10.6342/NTU201901446

Activation	$\phi(x)$	μ_k	Norm-preserving σ_w^2
Linear	x	1	1
ReLU	$[x]_+$	$1/2$	2
Hard Tanh	$[x+1]_+ - [x-1]_+ - 1$	$erf\left(\frac{1}{\sqrt{2}\sigma}\right)$	$erf\left(\frac{1}{\sqrt{2}\sigma}\right)^{-1}$

Table 3.1: The μ_k, the k-th moments of series expansion of the moment generating function associated with activation functions, is provided in this table. The derivation is given by [27]. The rightmost column are the norm-preserving (σ_w^2), which will be further discussed in 4.2. Note that the σ^2 is the variance of the hidden nodes. It shows that the maximal depth of ReLU activation is much less than linear and tanh activations, and that the orthogonal weights and the residual architecture can improve the maximal depth of networks.

Random Matrix W	s_1
Gaussian	-1
Orthogonal	0

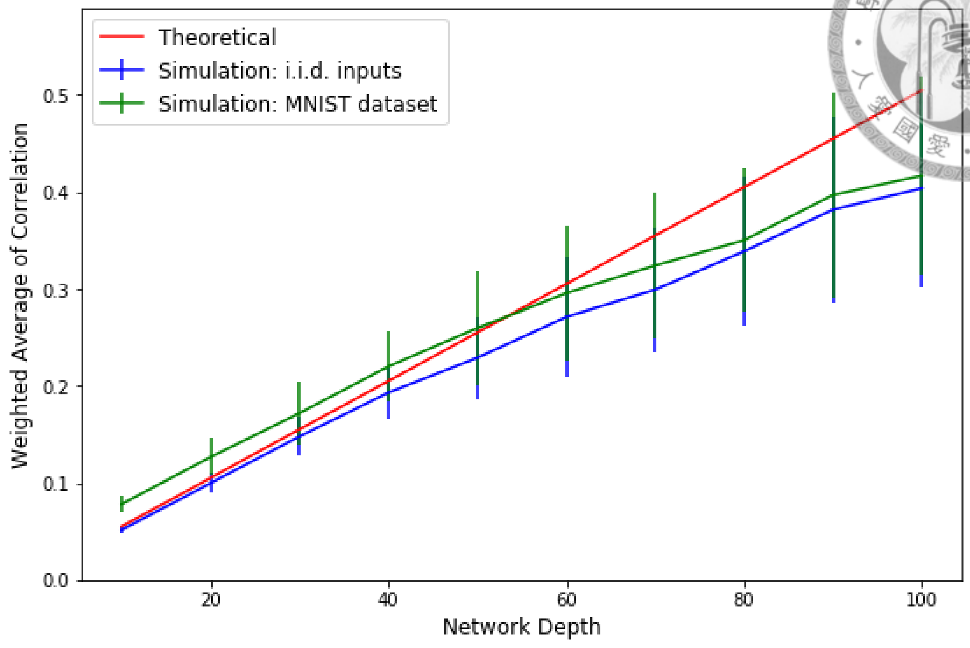
Table 3.2: The s_1 of different activation functions. The derivation is given by [27]. Note that the s_1 is invariant to the matrix scale. That is, the results are also suitable for the scaled-Gaussian and the scaled-orthogonal initializations.

Architecture	Weight initialization.	Activation	Maximal depth
Feed-forward network	Scaled Gaussian	tanh	4243
		linear	2606
		relu	262
	Orthogonal	tanh	> 10000
		linear	> 10000
		relu	325
Residual network	Scaled Gaussian	tanh	> 10000
		linear	> 10000
		relu	> 10000

Table 3.3: The maximum depth for representation power. The maximum depth is defined as the maximal L such that in less than half of 20 runs(i.e. 10 runs), the approximated number of effective nodes achieve greater than 1.

doi:10.6342/NTU201901446

(a) Network width $N = 200$



(b) Network width $N = 500$

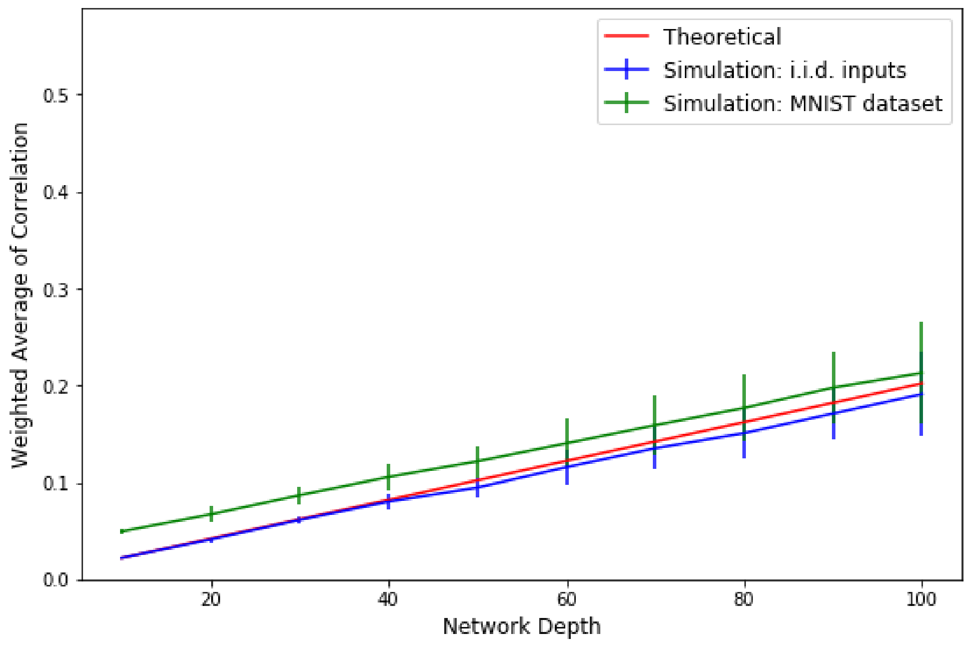


Figure 3.5: The results of VNI R_{sq} with respect to network depth L for the network width 200 and 500. The red line is calculated from eqn. (3.8), the blue line is computed from eqn. (3.5) with the input data of zero mean and i.i.d input data, and the green line is computed from eqn. (3.5) with MNIST data. The VNI R_{sq} expressed in eqn. (3.8) is very close to the original definition in eqn. (3.5).

doi:10.6342/NTU201901446

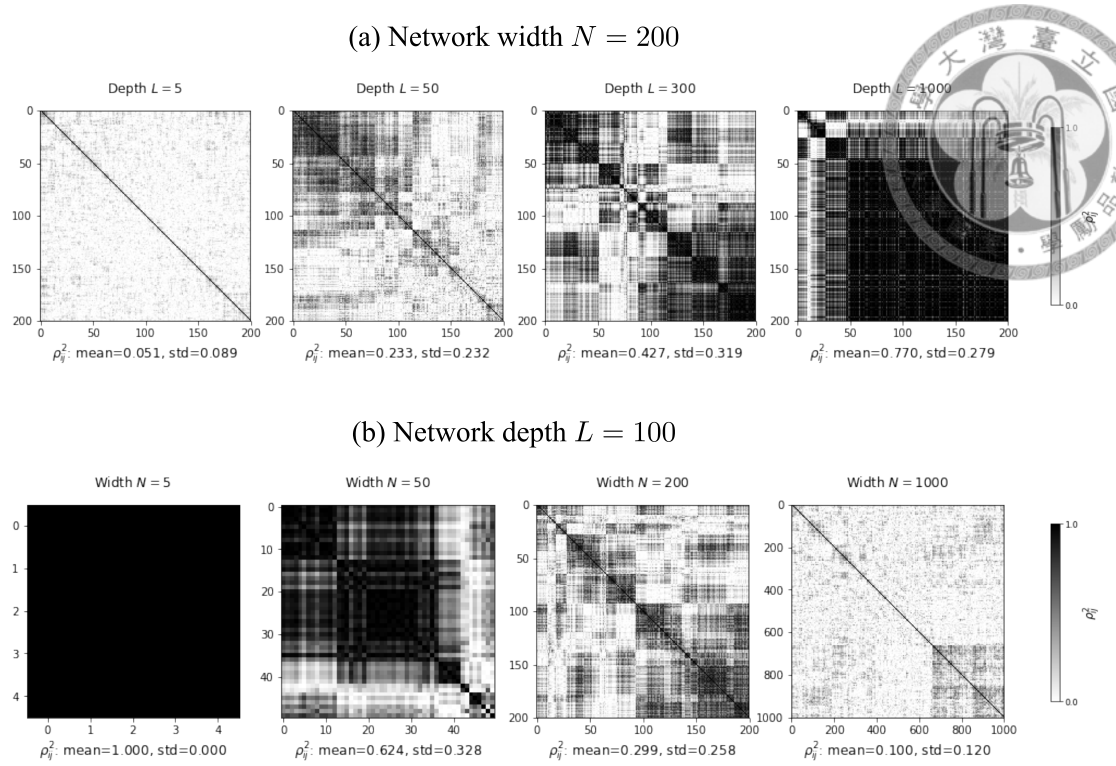


Figure 3.6: The magnitudes of correlation coefficient ρ_{ij} between output nodes. The black color means $\rho_{ij}^2 = 1$ while the white color indicates $\rho_{ij}^2 = 0$. The top row shows that the correlation is positive related to the network depth L, and the bottom row presents that the correlation is negatively related to the network width N. Note that we rearrange the node index to cluster the correlated nodes.

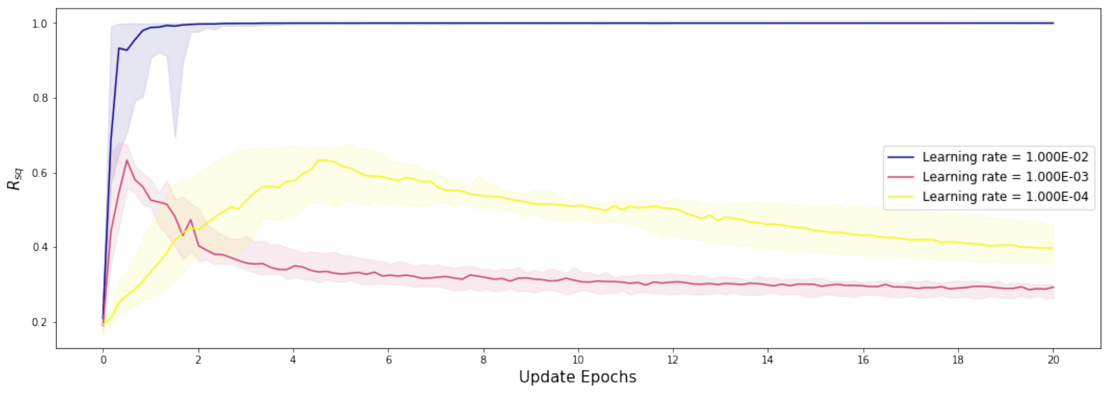
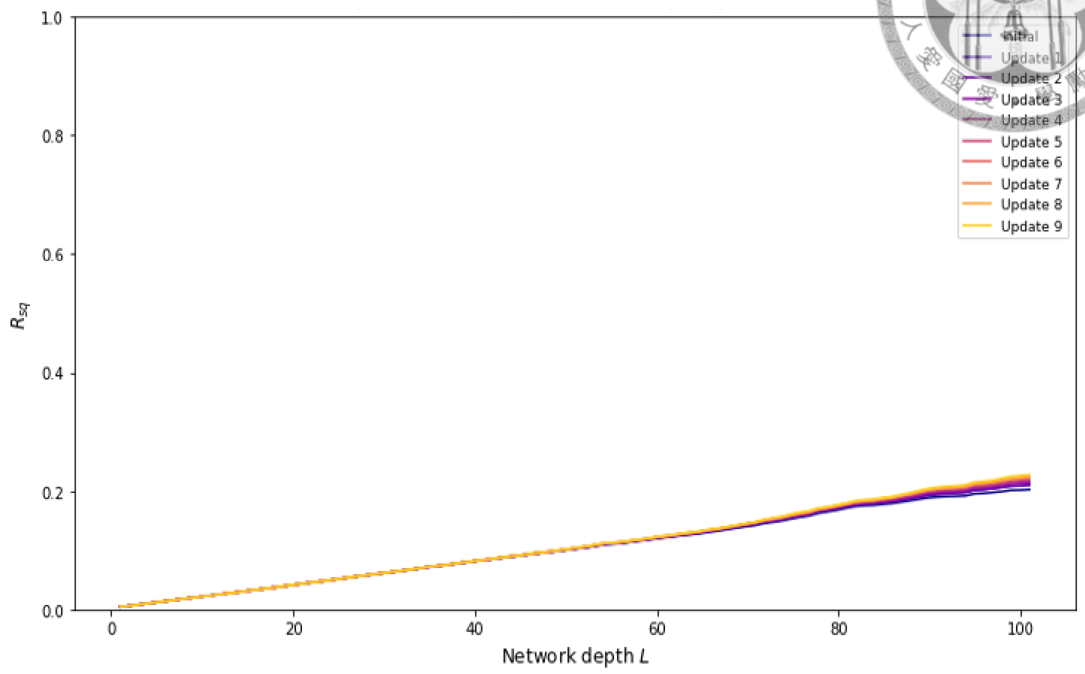


Figure 3.7: The dynamics of VNI R_{sq} of the output layer. The training is performed on the MNIST dataset 20 times, and then we evaluate the quartiles of the output VNI R_{sq} for different learning rates. Severe intensification of VNI (increases to 1) may occur as shown by the blue line which is trained with the learning rate of 10^{-2}. Otherwise VNI rises initially, and then decreases to a value which is larger than the initial VNI.

doi:10.6342/NTU201901446

(a) Learning rate $= 10^{-3}$



(b) Learning rate $= 10^{-2}$

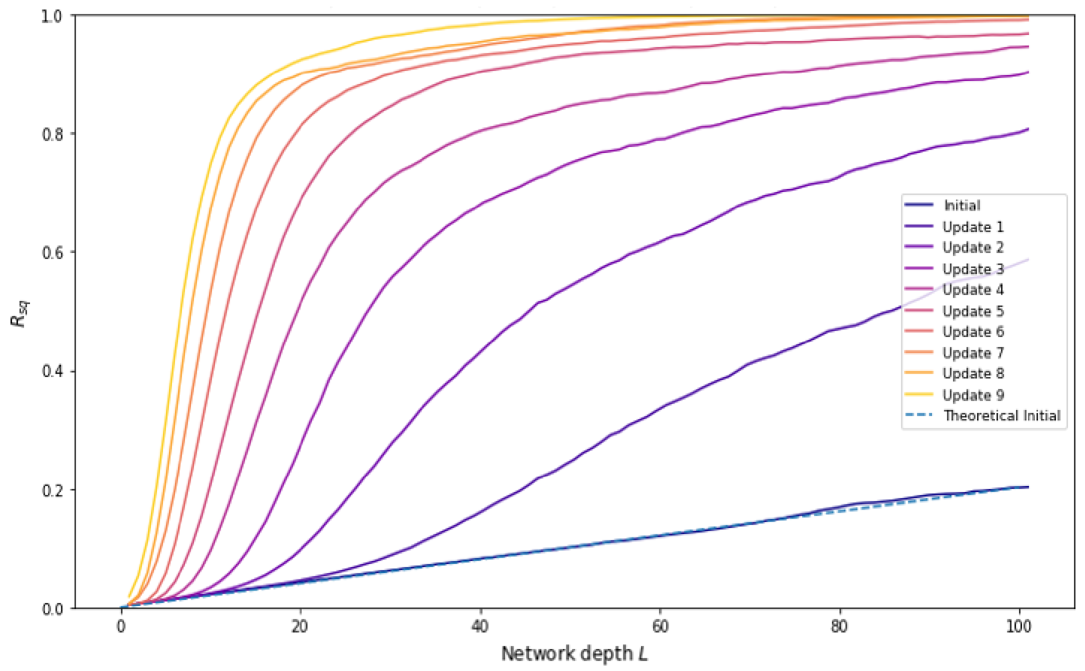
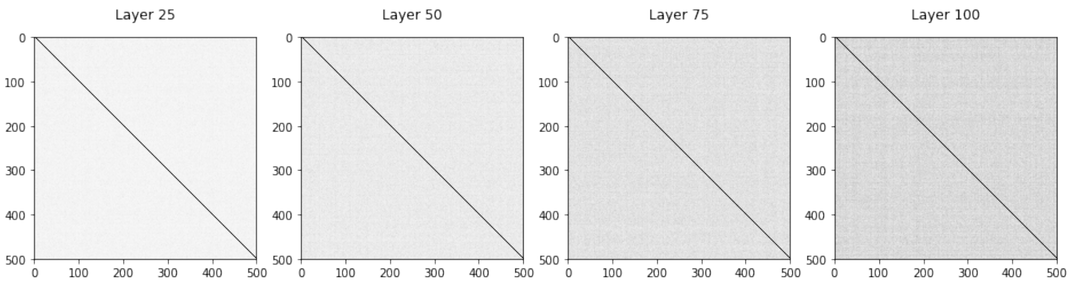


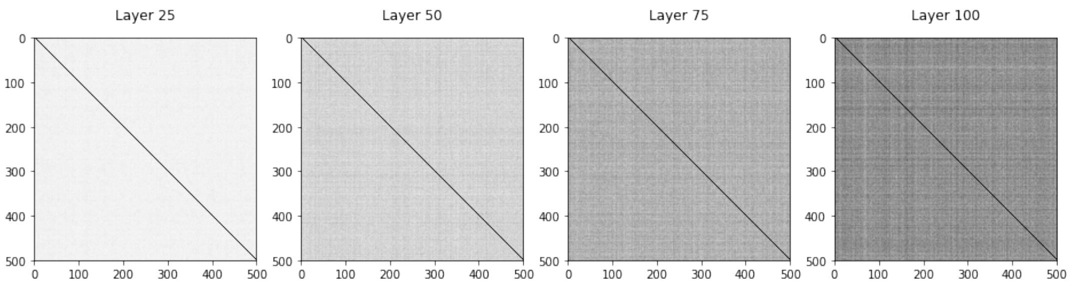
Figure 3.8: The dynamics of VNI R_{sq} of hidden layers. The training is performed on the MNIST dataset 50 times, and then we evaluate the averages of hidden layer VNI R_{sq} for learning rates $\in \{10^{-3}, 10^{-2}\}$.

doi:10.6342/NTU201901446

(a) Initial



(b) After 5 updates



(c) After 10 updates

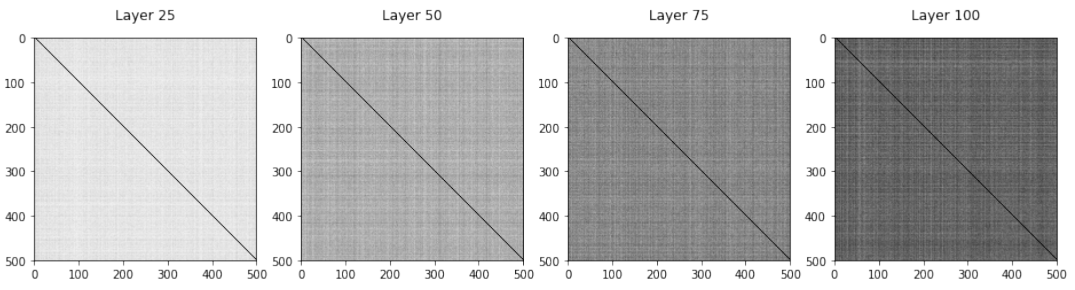


Figure 3.9: The averages of squared correlation coefficients ρ_{ij}^2 over 50 runs. It presents that overall, the correlation of each hidden layer are highly intensified.

doi:10.6342/NTU201901446

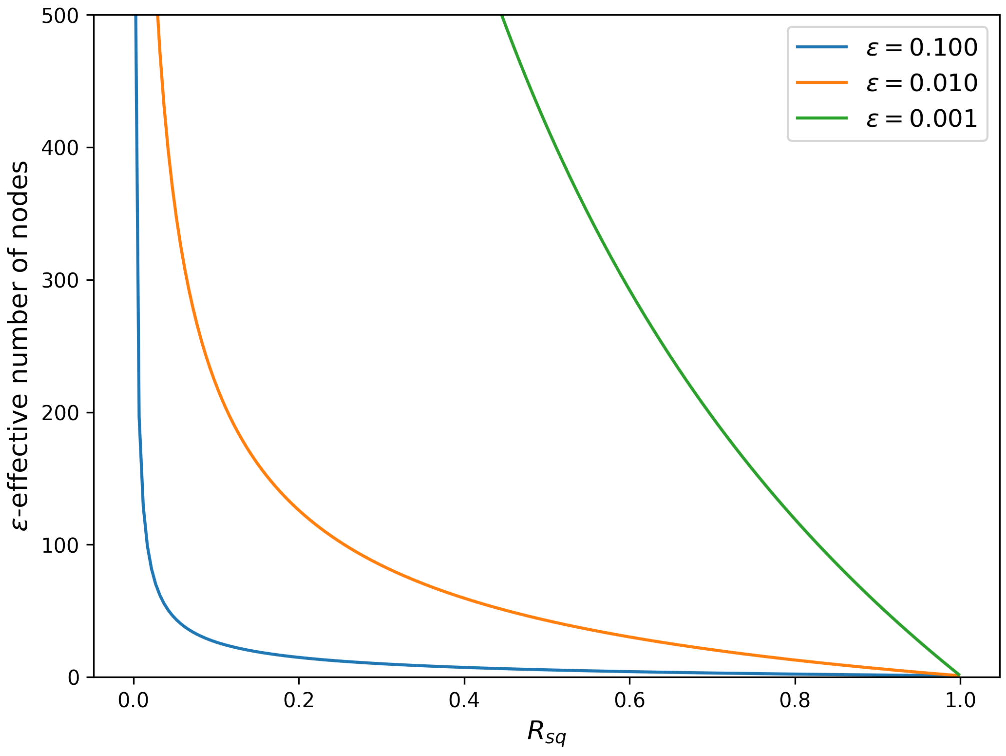


Figure 3.10: The numerical solution of the effective number of nodes for given ε. It can be observed that the effective number of nodes vanishes as the VNI R_{sq} increases to 1. Also, if a stricter (i.e. larger) ε is chosen, the ε-ENN vansihes much faster. Note that the number of random variables N is set to 500.

doi:10.6342/NTU201901446

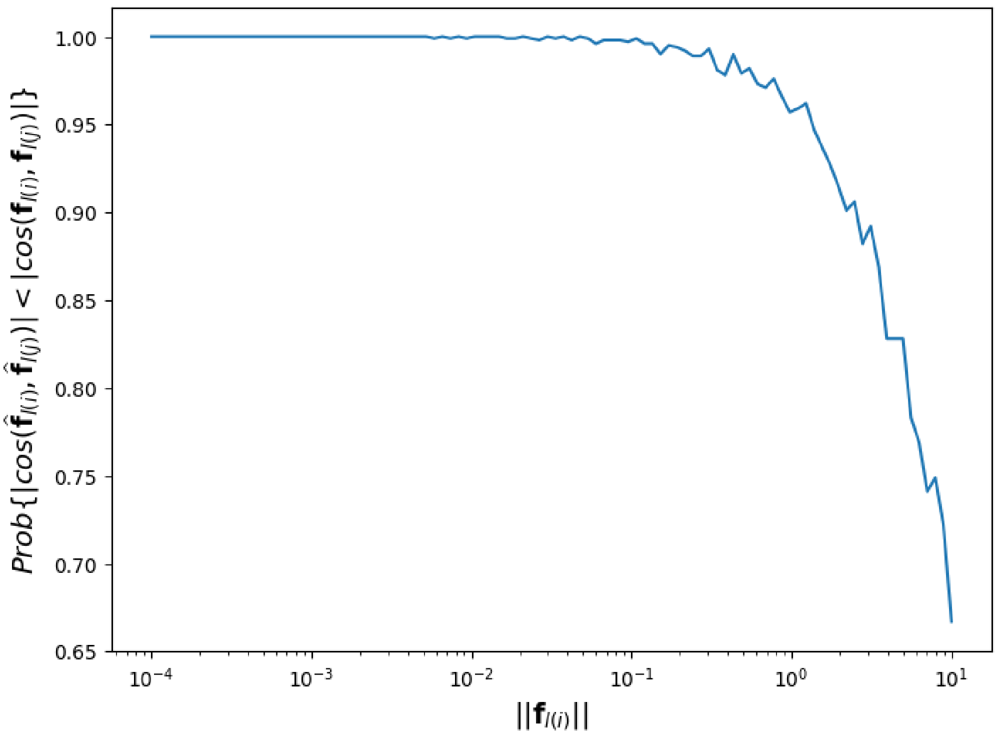
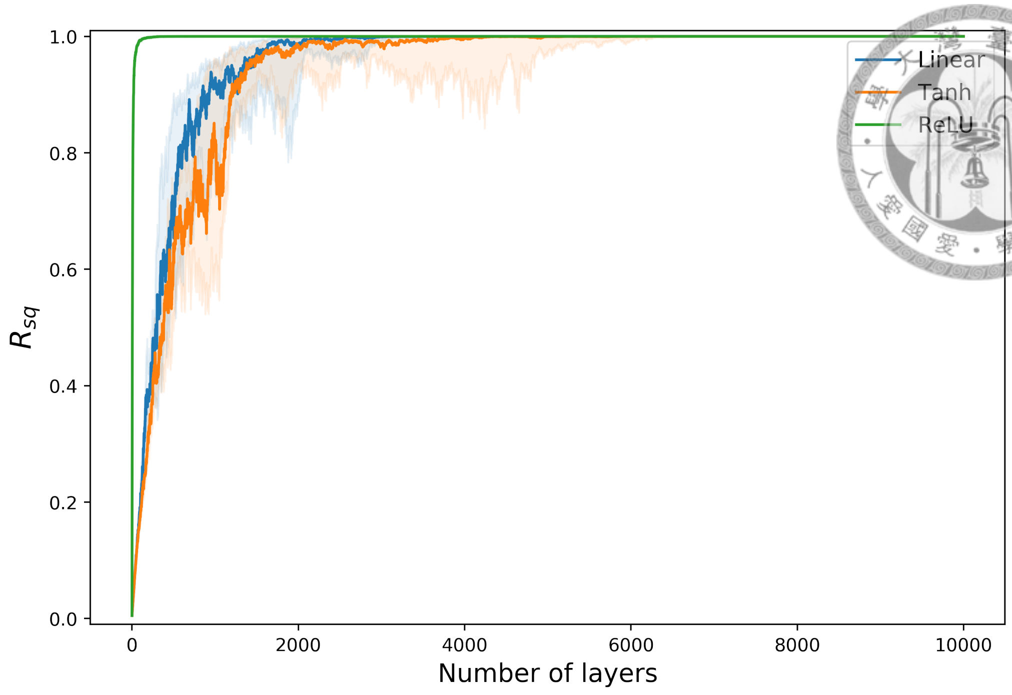


Figure 3.11: The probability of $|cos(\widehat{\mathbf{f}}_{l(i)}, \widehat{\mathbf{f}}_{l(j)})| < |cos(\mathbf{f}_{l(i)}, \mathbf{f}_{l(j)})|$ is shown in this figure. The simulation is performed with 1000 pairs of random vectors drawn from zero-mean white Gaussian for every $||\mathbf{f}_{l(i)}||$. The network width N is set to 500. Note that practically, $||\mathbf{f}_{l(i)}||$ is much smaller than 1 because the normalization operation (e.g. batch normalization [20]) included in $\mathbf{F}_l$ will reduce the magnitude of nodes for the residual network. It shows that with high probability, the residual-like transformation is closer to the orthogonality than the vanilla feed-forward transformation.

doi:10.6342/NTU201901446

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



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

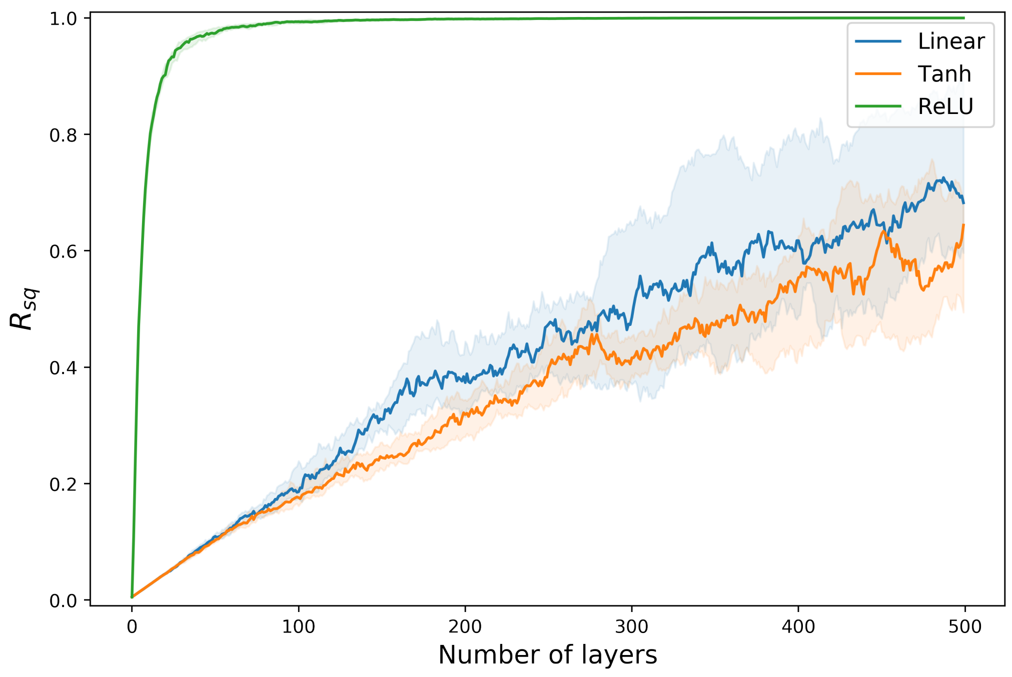
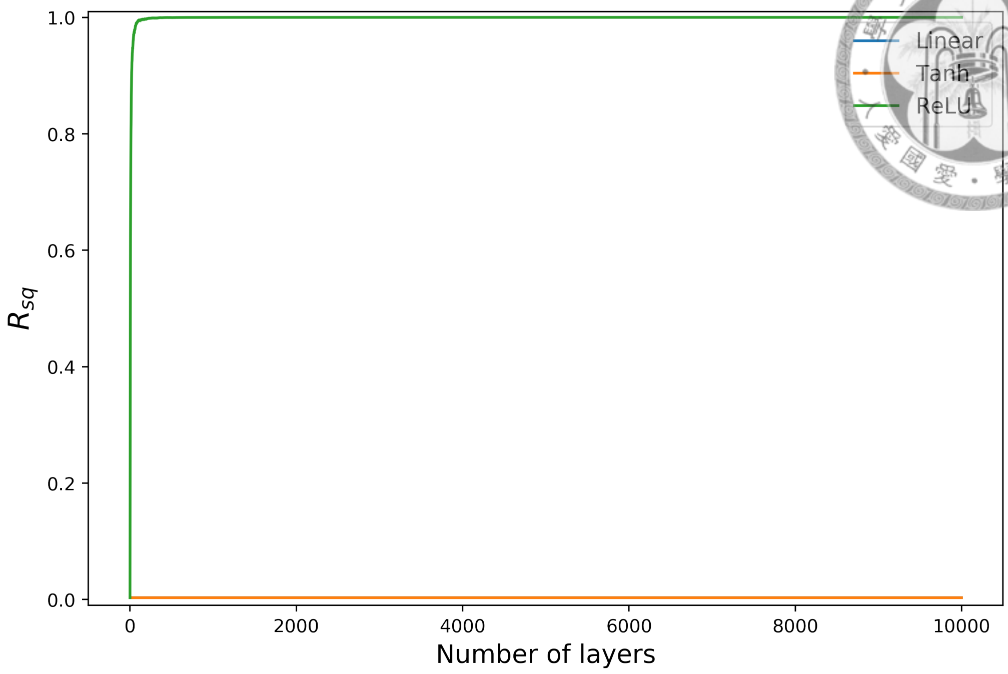


Figure 3.12: The network width N is set to 500 and the network depth L ranges from 1 to 10000. The VNI R_{sq} are evaluated according to the definition in eqn. (3.5). The weights are initialized with scaled-Gaussian distribution ([9, 16]). The activation functions of the networks include tanh, ReLU and linear. The simulation is repeated for 20 times, and the medians of the VNI over 20 runs are plotted as the solid lines, and the boundaries of the colored regions are the first and the third quartiles of the VNI. It is shown that for a feed-forward architecture under a non-orthogonal initialization, the initial VNI R_{sq} increases to 1 as L gets larger, and that the VNI of ReLU activation, especially, grows in the steepest with the network depth.

doi:10.6342/NTU201901446

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



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

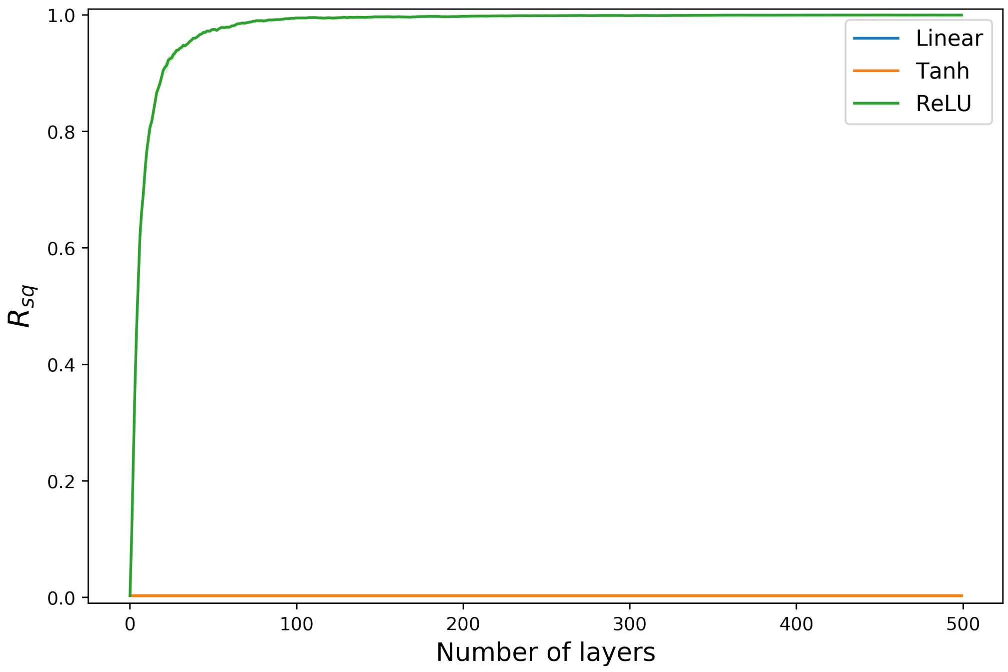


Figure 3.13: Similar settings to Figure 3.12 are used excluding the weights, which are initialized with scaled-Orthogonal distribution ([31]). It is shown that for a feed-forward architecture under the orthogonal initialization with linear or tanh activation function, the initial VNI R_{sq} remain nearly minimum (1/N) even when the network depth L gets larger, and that the VNI of ReLU activation increases to 1 as the network goes deeper.

doi:10.6342/NTU201901446

(a) Residual deep network with 1-layer skip (b) Residual deep network with 2-layer skip

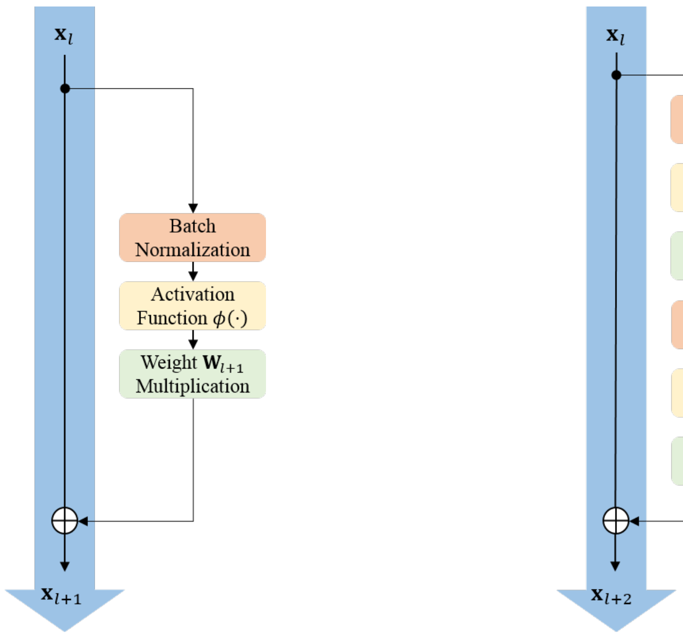
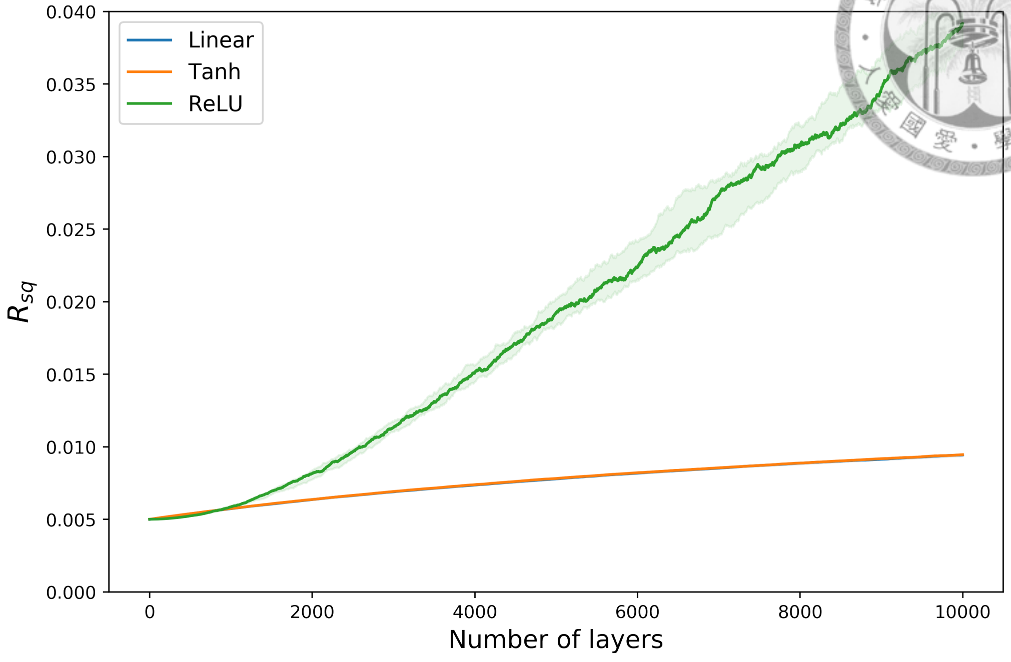

$\mathbf{x}_l$
Batch Normalization
Activation Function $\phi(\cdot)$
Weight $\mathbf{W}_{l+1}$ Multiplication
$\mathbf{x}_{l+1}$
$\mathbf{x}_l$
Batch Normalization
Activation Function $\phi(\cdot)$
Weight $\mathbf{W}_{l+1}$ Multiplication
Batch Normalization
Activation Function $\phi(\cdot)$
Weight $\mathbf{W}_{l+2}$ Multiplication
$\mathbf{x}_{l+2}$

Figure 3.14: The architectures of residual deep networks is presented in this figure. The network is composed of 4 operations: batch normalization ([20]), activation function, weight multiplication and the addition with the identity mapping from the skip connection. The left architecture has the 1-layer skip, and the right one has the 2-layer skip, which is the original definition in previous works([18]).

doi:10.6342/NTU201901446

(a) 1-layer shortcut defined in Figure 3.14a



(b) 2-layer shortcut defined in Figure 3.14b

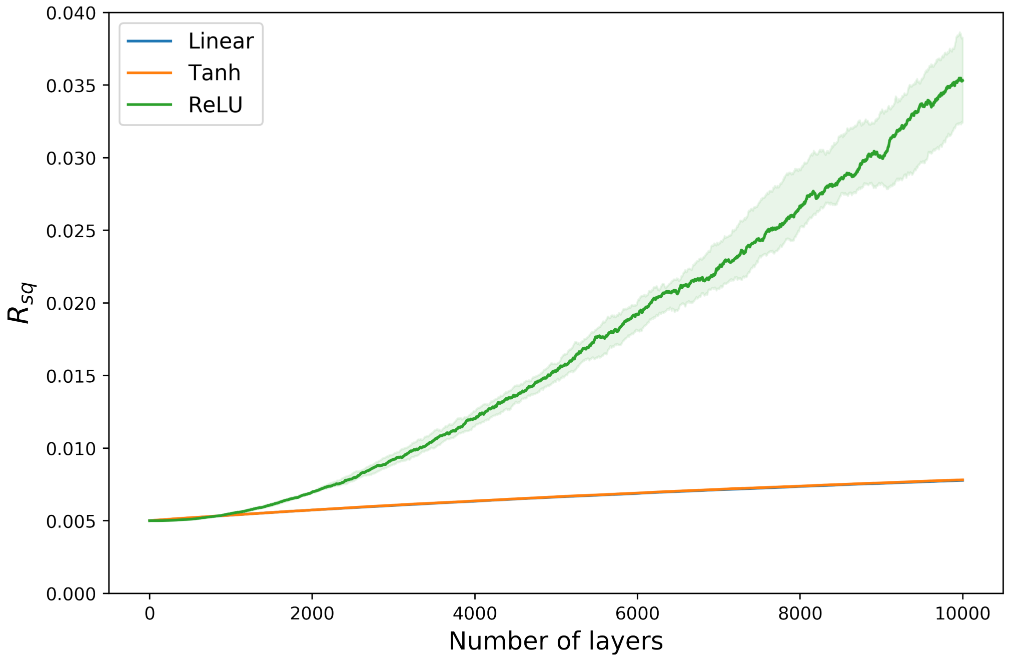
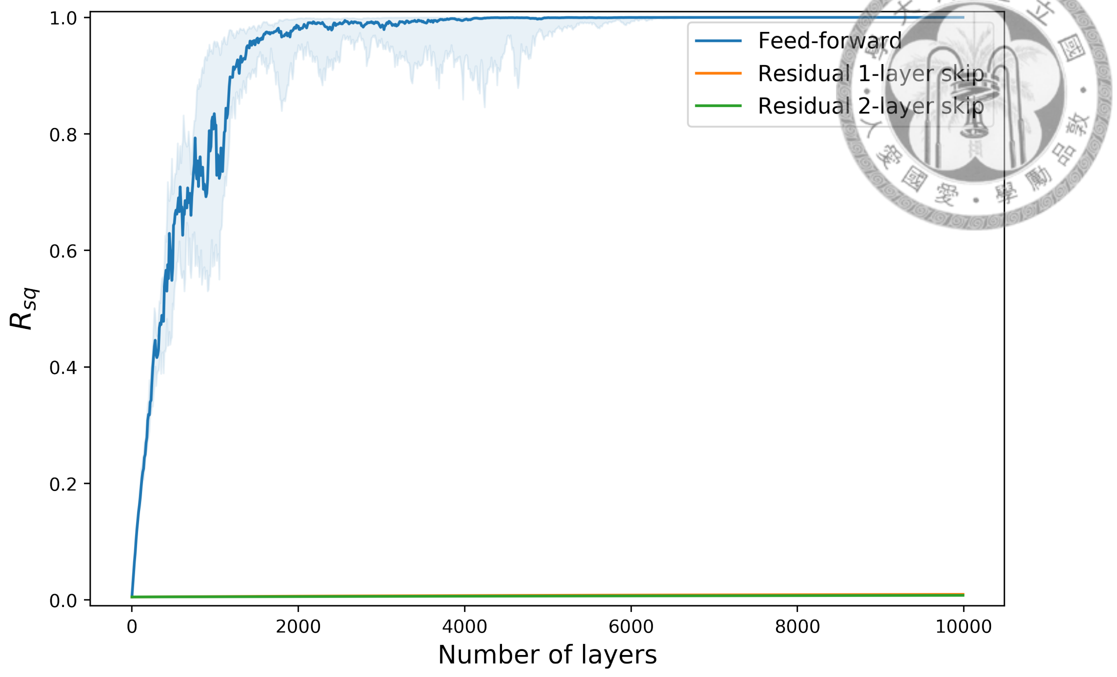


Figure 3.15: Similar settings to Figure 3.12 are used excluding the network architectures, which are defined in Figure 3.14. It is shown that for a residual-like architecture, the VNI R_{sq} grows slowly as the network depth L gets larger. Note that for the 2-layer shortcut architecture, we only perform the simulation with even numbers of hidden layers.

doi:10.6342/NTU201901446

(a) Tanh activatoin function



(b) ReLU activation function

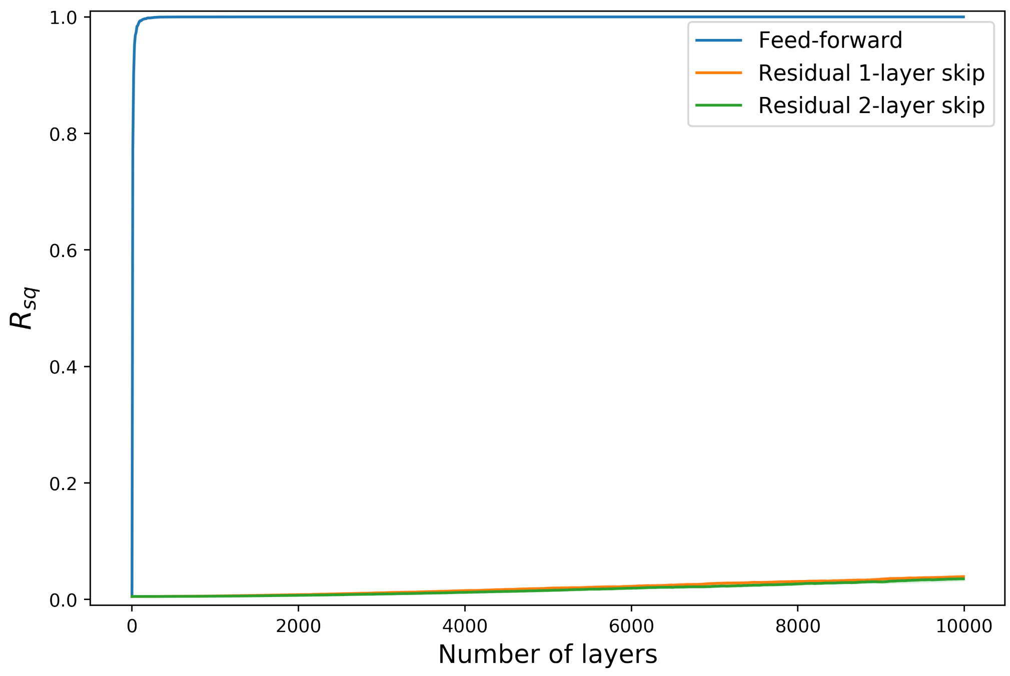


Figure 3.16: To compare the VNI R_{sq} of different network architectures, we plot the curves of Figure 3.4 and 3.6 in the same graph. The weight initialization methods are set to the same (Gaussian distribution). We can see that for both tanh and ReLU cases, the residual architecture has much smaller VNI R_{sq} when the network depth L increases. It implies that the residual skip connections can keep more representation power when a very deep network is considered.

doi:10.6342/NTU201901446

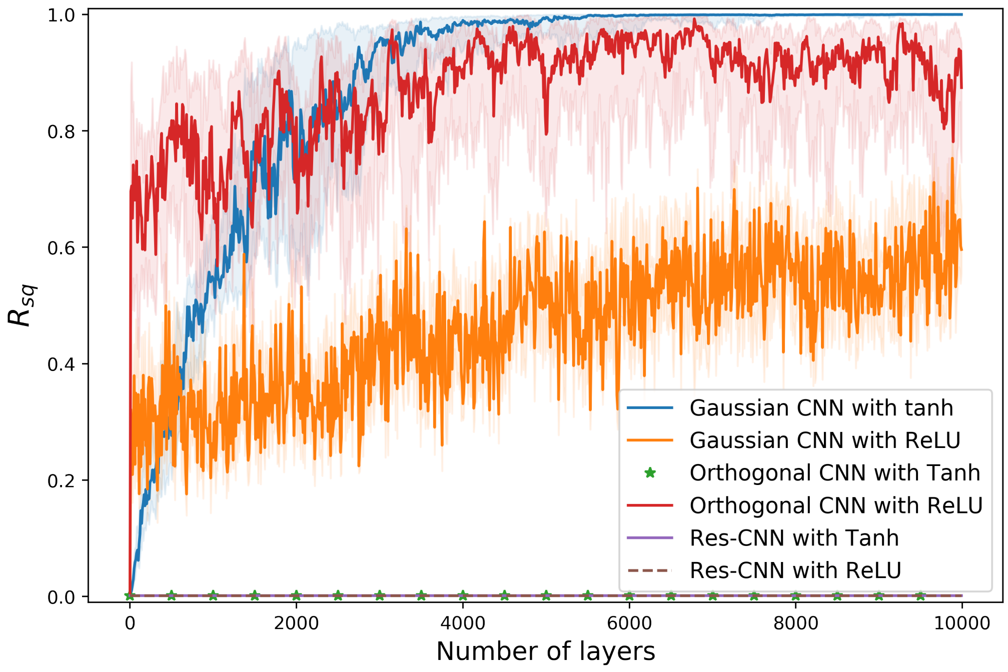


Figure 3.17: To compare the VNI R_{sq} of different network architectures of convolutional neural networks, we plot the results of architectures with and without residual connection. The weight initialization methods are set to the same (Gaussian distribution). Similar to Figure 3.16, we can see that for both tanh and ReLU cases, the residual architecture has much smaller VNI R_{sq} when the network depth L increases. Also, compared with the feed-forward deep neural network in Figure 3.16, the VNI R_{sq} increases slower with respect to the network depth L. It implies that the residual skip connections and the convolution operation can keep more representation power when a very deep network is considered.

doi:10.6342/NTU201901446

Chapter 4

Variance propagation of deep neural networks

4.1 Comparison of exploding/vanishing gradients and vanishing nodes

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 [9], 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

doi:10.6342/NTU201901446

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. (3.4), 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

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

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

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

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 from eqn. (3.1) we have $\frac{\partial Cost}{\partial \mathbf{W}_l} = \mathbf{x}_{l-1} \cdot \frac{\partial Cost}{\partial \mathbf{h}_l}$, 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

doi:10.6342/NTU201901446

weight gradient can be evaluated as

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

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. (3.8) 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. (3.8) 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

In the Section 4.1, it is suggested that $(\sigma_w^2\mu_1) = 1$ is the condition to prevent the vanishing/exploding gradients problem. Note that the parameter μ_1 is decided by the activation function of the network, and the other parameter σ_w^2 can be controlled by the scale of the initial weight. That is, if the weight matrices are correctly initialized, the gradient flow will neither vanish nor explode as the network depth L increases. We call this kind of weight initialization method a *"norm-preserving weight initialization"*.

In Table 3.1, the norm-preserving σ_w^2 are provided. Note that in Section 3.1, the variances of weight matrices are defined as σ_w^2/N. Therefore the norm-preserving weight variance of ReLU activation, for example, is $2/N$. For the Gaussian distribution, we

doi:10.6342/NTU201901446

can simply set the mean to zero and modify the standard deviation to meet the norm-preserving condition, and for the uniform distribution, the support should be set to $\left[-\sqrt{3\sigma_w^2/N}, \sqrt{3\sigma_w^2/N}\right]$ to achieve the zero-mean and the norm-preserving property. For the orthogonal initialization, the zero-mean is already achieved, and the L_2-norm of the orthogonal basis should be set to σ_w.

Another point needs to be considered is the network width N in the variances of weight matrices σ_w^2/N. When the widths of two adjacent layers are not the same, the N is suggested to be $(N_l + N_{l-1})/2$ as a compromise between forward and backward variance propagation ([9]). Also in [16], the N for a convolutional layer is suggested to be k^2c, where k denotes the kernel size and c represents the number of channels.

It is worth mentioning that for the residual-like architectures, the norm-preserving weight initialization designed for feed-forward networks ([9, 16]) are not the optimal initialization methods. Instead, the optimal initialization method should depend on the network depth L ([42]) to prevent exploding values of nodes. The norm-preserving σ_w^2 of a residual architecture is suggested to be $\mathcal{O}(L^{-1})$ to avoid explosion ([36]). However, the residual-like architecture is often combined with batch normalization ([20]), which reduces the importance of weight initialization.

4.3 The two obstacles for training deep nerual networks

In Figure 4.1, we provide a schematic diagram for evaluating architectures of deep neural networks. The network depth L, the network width N, the weight initialization scale σ_w^2, the moments associated with weight s_1 and the moments associated with activation μ_k are taken into consideration. For the horizontal axis, we use the metric $(\sigma_w^2\mu_1)$ to determine whether a network will explode or vanish when the depth L goes deeper. If a network has $(\sigma_w^2\mu_1) = 1$, then its scales of gradients will neither vanish nor explode even if the depth L becomes larger. Otherwise, the further $(\sigma_w^2\mu_1)$ is from 1, the more severe gradients will exponentially explode/vanish. For the vertical axis, we take R_{sq} as the metric. From eqn. (3.8), we know that R_{sq} is decided by L, N, μ_k and s_1. If R_{sq} can reach exactly $1/N$, then by eqn. (3.8), we can show that correlations of output layer nodes will never

doi:10.6342/NTU201901446

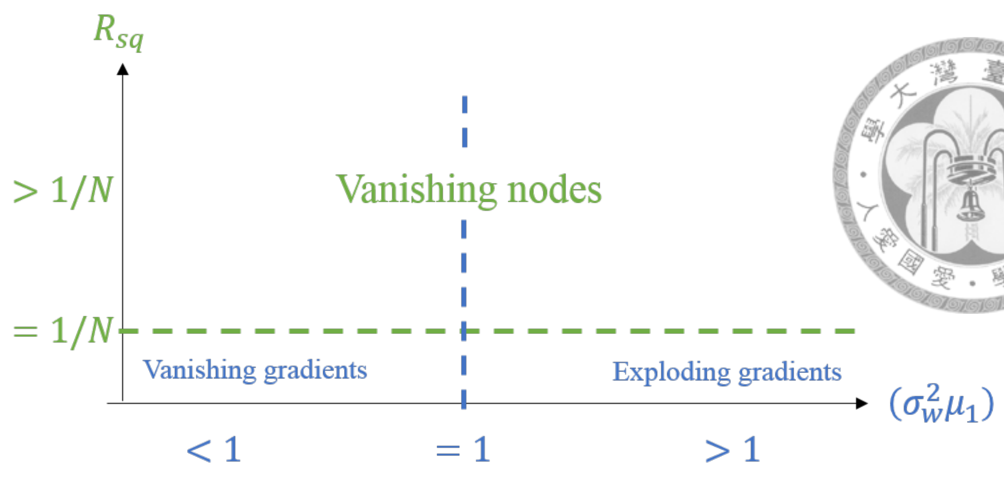


Figure 4.1: The schematic diagram for deep neural network architectures. To avoid the deep network gradients from exploding or vanishing, it is suggested that the condition $(\sigma_w^2 \mu_1) = 1$ should be met. For vanishing nodes problem, the VNI R_{sq} of the network shall not increases to 1 as the network depth L goes deeper, otherwise the representation power of the network will be insufficient for the training task. That is, to overcome the two obstacles of training a very deep network, the best network setting is located at the intersection of vertical and horizontal dashed line.

accumulate even if the depth L increases. Therefore by Figure 4.1, we can determine whether a neural network with specific parameters will suffer from exploding/vanishing gradients and vanishing nodes.

For example in previous works on ultra-deep neural networks, [41] chose an activation function that has $\mu_2/\mu_1^2 \approx 1$ and initialized the weights via appropriately-scaled orthogonal matrices ([31]) which have $s_1 = 0$ (from [26]) and $(\sigma_w^2 \mu_1) = 1$. Hence the VNI R_{sq} of the network will reach $1/N$, which will not accumulate as the depth L increases according to eqn. (3.7). The parameter setting of the network is located at the intersection of vertical and horizontal dashed line in Figure 4.1, and thus the network does not suffer from vanishing/exploding gradients and vanishing nodes.

According to eqn. (3.7), if R_{sq} gets larger, then m_2 is also larger with respect to m_1^2. Recall that m_i is the i-th moment of eigenvalues of $\mathbf{J}\mathbf{J}^T$, so the variance of eigenvalues of $\mathbf{J}\mathbf{J}^T$ is $m_2 - m_1^2$. Also, eigenvalues of $\mathbf{J}\mathbf{J}^T$ is equivalent to the squared singular values of $\mathbf{J}$. Thus, if the variance of eigenvalues of $\mathbf{J}\mathbf{J}^T$ is too big, the Jacobian $\mathbf{J}$ will become ill-conditioned. Therefore, we can relate R_{sq} to the condition number of Jacobian $\mathbf{J}$, and thus we can link the vanishing node problem to the ill-conditioned Jacobian, which is

doi:10.6342/NTU201901446

emphasized in previous works ([26, 27, 31]). Moreover, *dynamical isometry*, a stronger condition for deep neural networks, is described as ”*all* singular values of the Jacobian concentrate near 1” by [26, 31]. That is, if *dynamical isometry* is achieved, then the variance of singular values of the input-output Jacobian will approach nearly zero, which implies $m_2 - m_1^2 \approx 0$. Therefore, the R_{sq} will also remain nearly $1/N$ even at a large depth L. Therefore, we can say that dynamical isometry is not only related to the *learning speed* ([31]), but also linked to the node correlation R_{sq}, which is closely connected with the *learning capability* and the *representation power* of a deep neural network.

doi:10.6342/NTU201901446

Chapter 5

Experiments

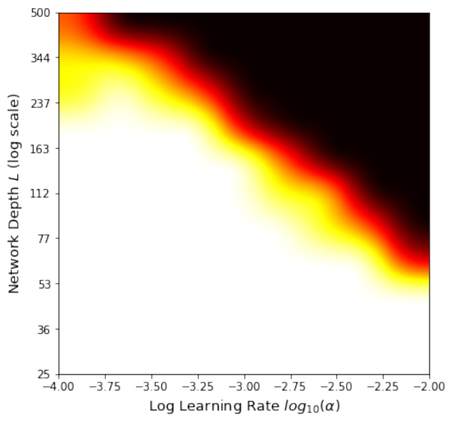
5.1 Probability of failed training caused by vanishing nodes

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 [23]. 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 3.7. 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[5].

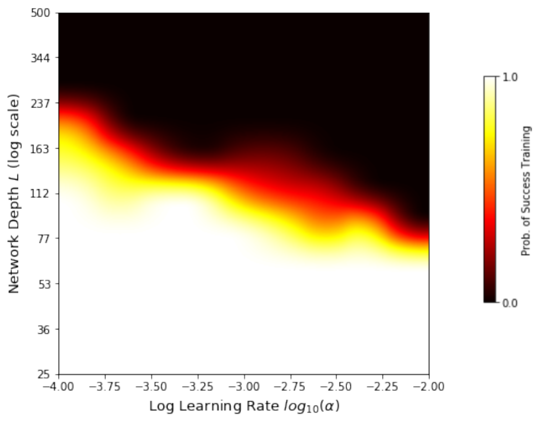
Figure 5.1 shows the results of two different activation functions (Tanh/ReLU) with two different weight initializations (scaled-Gaussian/orthogonal from [31]). When a network with tanh activation functions is initialized with orthogonal weights, the term of $(\mu_2/\mu_1^2 - 1 - s_1)$ in eqn. (3.8) 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-

doi:10.6342/NTU201901446

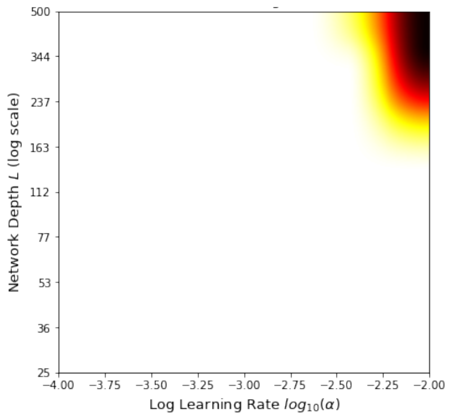
(a) Probability of Success (Tanh, Scaled Gaussian Init.)



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



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



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

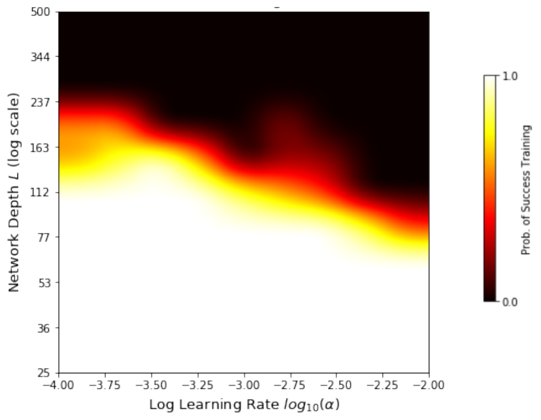


Figure 5.1: Probability of successful training for different network depth L and learning rate α (the SGD optimizer). The black color denotes zero probability of successful training.

doi:10.6342/NTU201901446

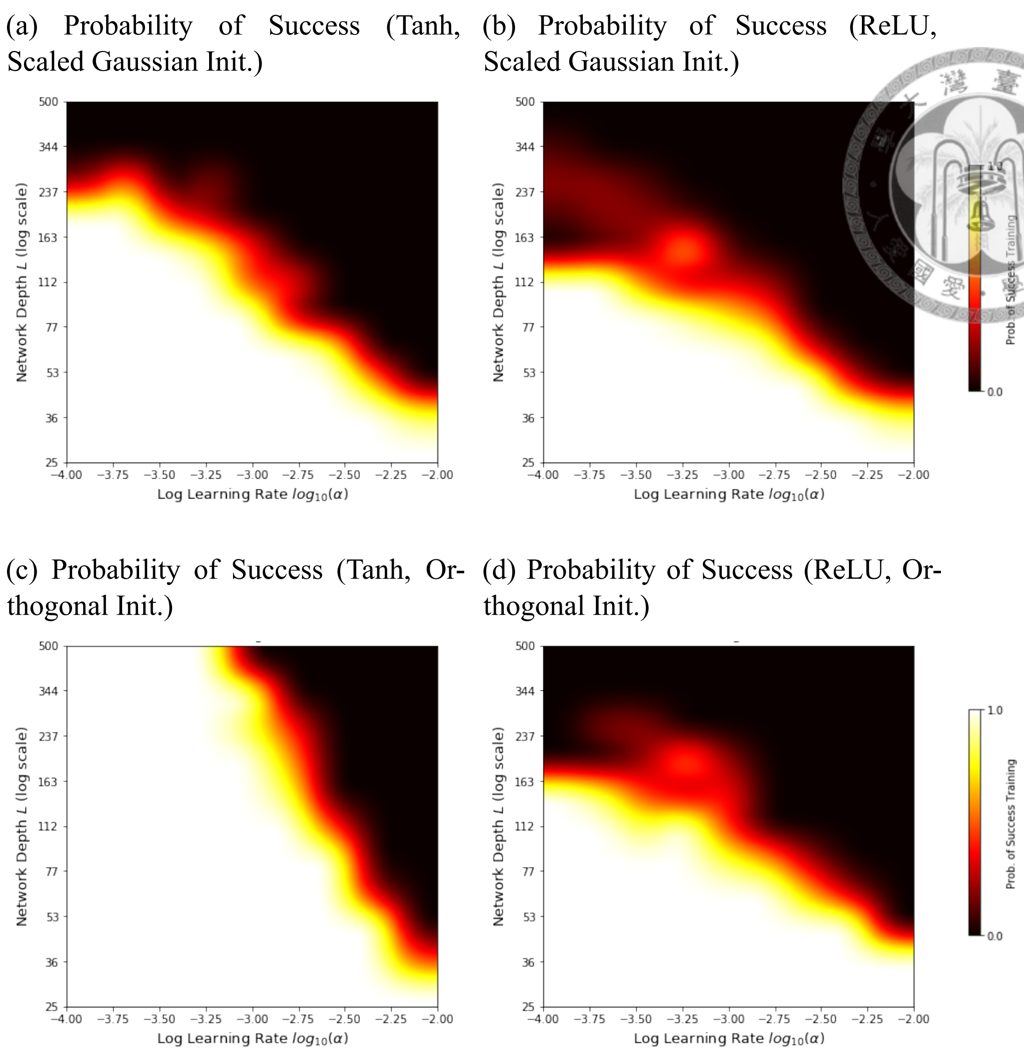
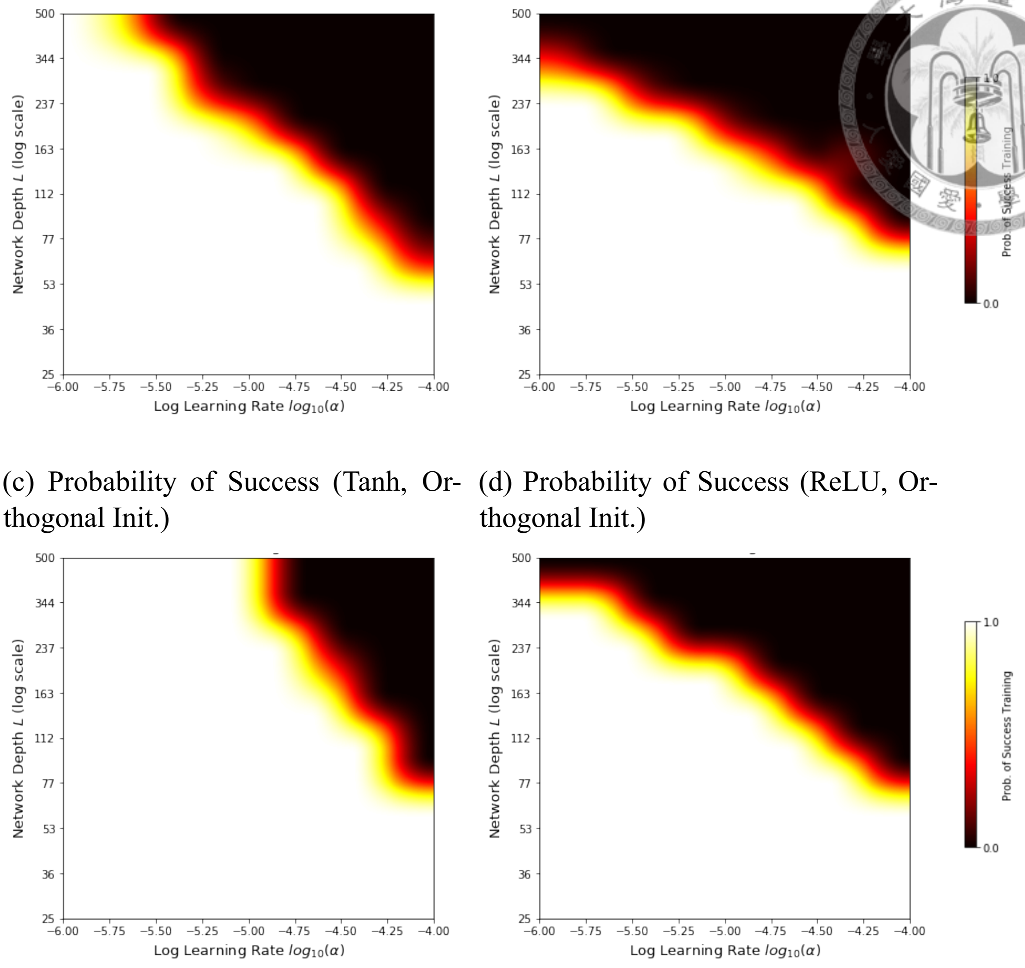


Figure 5.2: Probability of successful training for different network depth L and learning rate α (the SGD + Momentum optimizer). The networks are initialized with scaled Gaussian/orthogonal weights with Tanh/ReLu activation functions.

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

doi:10.6342/NTU201901446



(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.)

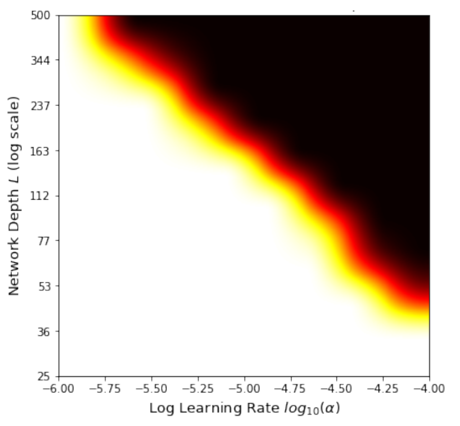
Figure 5.3: Probability of successful training for different network depth L and learning rate α (the Adam optimizer). The networks are initialized with scaled Gaussian/orthogonal weights with Tanh/ReLu activation functions.

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.

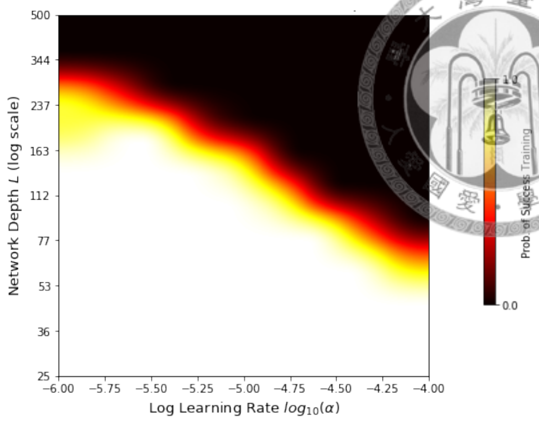
It is worth noting that, if more efficient optimization methods (e.g. Adam, RMSProp) are used, the feasible learning rate should become smaller. We can see that the boundary in Figure 5.2 has the offset to the left about 0.5 log unit comparing with Figure 5.1, and that in Figure 5.3 and 5.4 has the offset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

doi:10.6342/NTU201901446

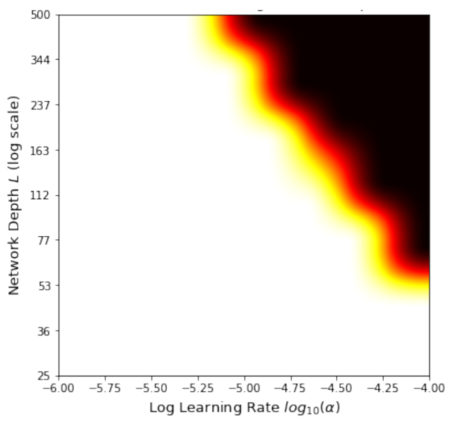
(a) Probability of Success (Tanh, Scaled Gaussian Init.)



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



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



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

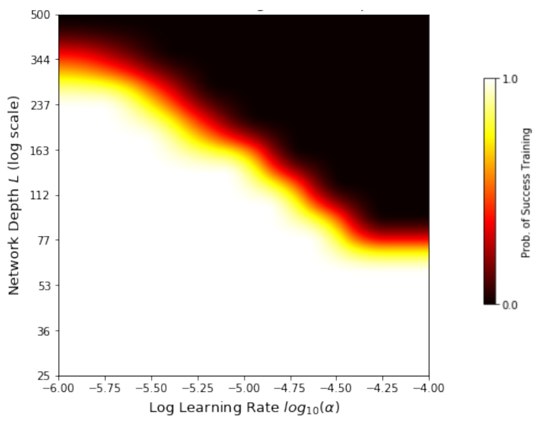


Figure 5.4: Probability of successful training for different network depth L and learning rate α (the RMSProp optimizer). The networks are initialized with scaled Gaussian/orthogonal weights with Tanh/ReLu activation functions.

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.

doi:10.6342/NTU201901446

5.2 Analyses of failed training caused by vanishing nodes

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.

doi:10.6342/NTU201901446

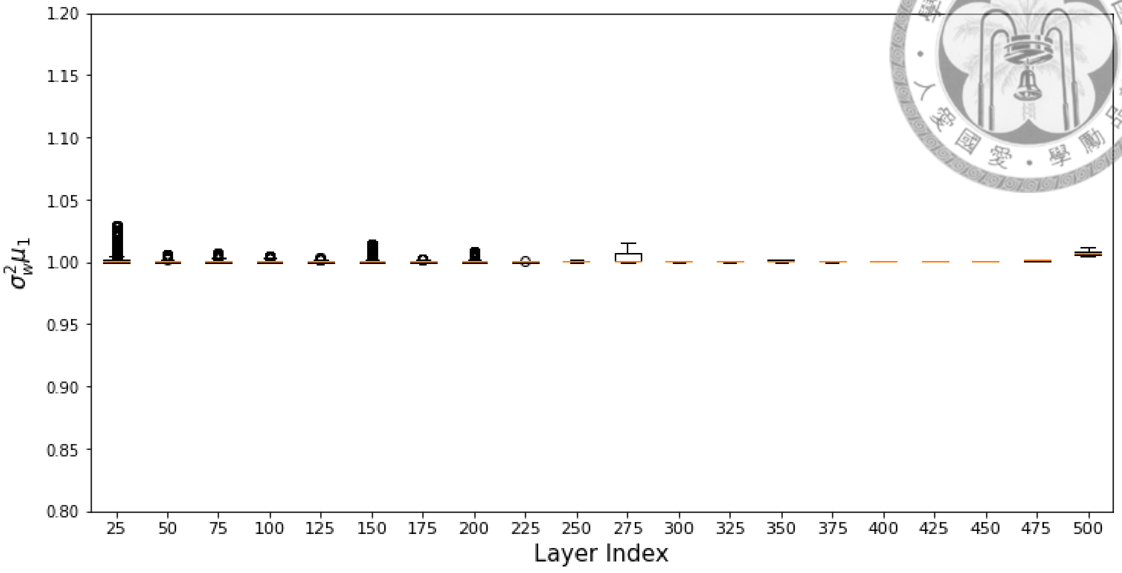


Figure 5.5: Box and whisker plot of $\sigma_w^2\mu_1$ for networks with successful training. There are 18,579 successful runs.

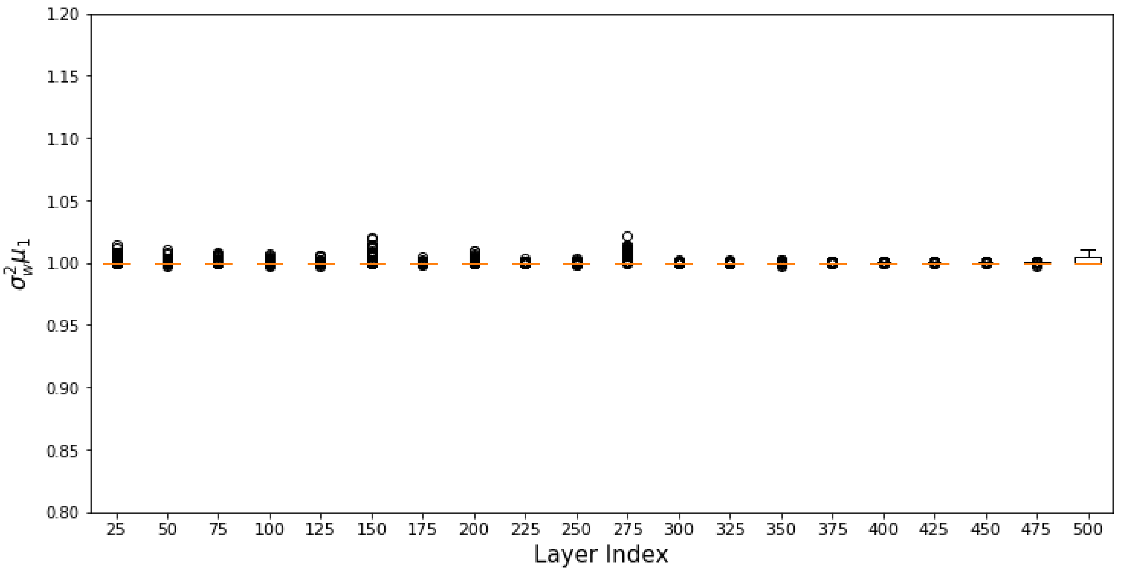


Figure 5.6: Box and whisker plot of $\sigma_w^2\mu_1$ for networks with failed training. There are 13,101 failed runs.

doi:10.6342/NTU201901446

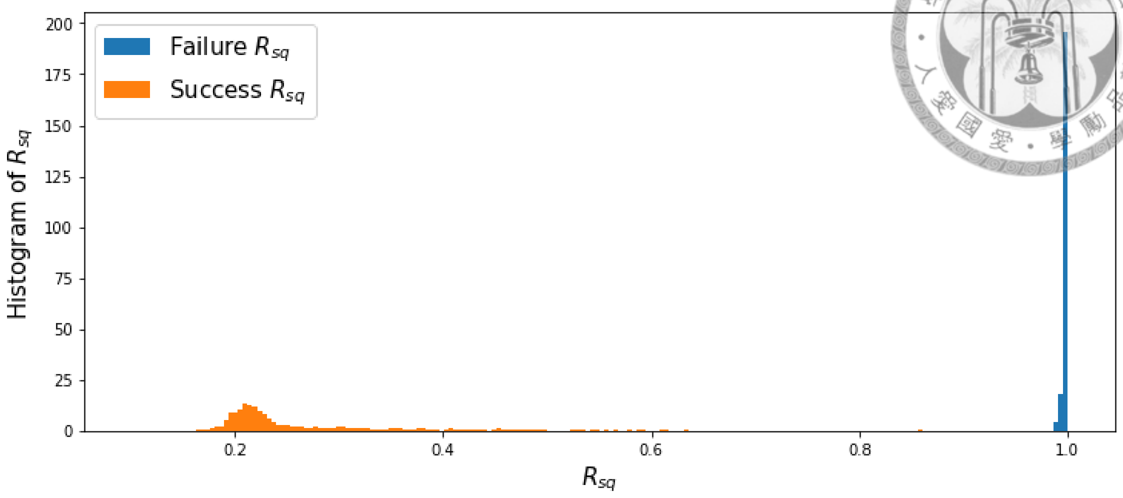


Figure 5.7: Histogram of R_{sq} of failed/successful networks.

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.

doi:10.6342/NTU201901446

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. Via this theoretical tool, we proof that the representation power of a network vanishes as the VNI goes to 1. The effective number of nodes goes to 1 as when the VNI equals to 1, which is called the ”*network collapsing*”. Also, we show that for a non-orthogonal initialized network, the VNI increases as the network depth gets larger, and it asymptotically goes to 1 as the network is very deep. That is, the network collapses when we consider a very deep feed-forward neural network.

However, if weight matrices are initialized with orthogonal distribution, or if a residual-like architecture is applied, then the network will not collapse at a large depth. We show theoretically that orthogonal weight can have small VNI at initial, and that the network with identity shortcut connection is closer to the orthogonality. Numerical simulations are also performed on different activation functions, weight initializations and network archi-

doi:10.6342/NTU201901446

tectures, which have a consistent result with our derivation. Both theoretical and numerical results suggest that the weight initialization and the architecture of a network determine its trainable depth. Orthogonal weight initializations and residual-like architectures, from this point of view, are relatively better for training a very deep neural network.

Moreover, we explore the difference between vanishing/exploding gradients and vanishing nodes, and suggest a criterion to predict the occurrence of two problems by the network depth, the network width, the activation, and the weight initialization. Finally, experimental results show that vanishing/exploding gradients and vanishing nodes are two different challenges that make training deep neural networks difficult.

doi:10.6342/NTU201901446

Bibliography

[1] M. Abramowitz. *Handbook of Mathematical Functions, With Formulas, Graphs, and Mathematical Tables*. Dover Publications, Inc., New York, NY, USA, 1974.

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

[3] J. L. Ba, J. R. Kiros, and G. E. Hinton. Layer normalization. *arXiv:1607.06450*, 2016.

[4] 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.

[5] 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.

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

[7] 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.

doi:10.6342/NTU201901446

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

[9] 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.

[10] X. Glorot, A. Bordes, and Y. Bengio. Deep sparse rectifier neural networks. *AISTATS*, 15:275, 2011.

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

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

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

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

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

[16] 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.

[17] 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.

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

doi:10.6342/NTU201901446

[19] 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.

[20] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *ICML*, 2015.

[21] J. R. Ipsen. Products of independent gaussian random matrices. *arXiv:1510.06128*, 2015.

[22] 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.

[23] 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.

[24] 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.

[25] V. I. Oseledets. Multiplicative ergodic theorem: Characteristic lyapunov exponents of dynamical systems. *Trudy MMO (in Russian)*, 19:179–210, 1968.

[26] 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.

[27] 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.

doi:10.6342/NTU201901446

[28] 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.

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

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

[31] 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.

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

[33] 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.

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

[35] 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.

[36] M. Taki. Deep residual networks and weight initialization. *arXiv:1709.02956*, 2017.

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

doi:10.6342/NTU201901446

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

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

[40] 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.

[41] 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.

[42] G. Yang and S. S. Schoenholz. Mean field residual networks: On the edge of chaos. *Advances in neural information processing systems*, pages 7103–7114, 2017.

doi:10.6342/NTU201901446