Training Very Deep Neural Networks: Rethinking the Role of Skip Connections

Oyebade K. Oyedotun[a,*], Kassem Al Ismaeil[a], Djamila Aouada[a]

[a]*Interdisciplinary Centre for Security, Reliability and Trust (SnT), University of Luxembourg, L-1855 Luxembourg*

Abstract

State-of-the-art deep neural networks (DNNs) typically consist of several layers of features representations, and especially rely on skip connections to avoid the difficulty of model optimization. Despite the proliferation of different DNN models that employ various forms of skip connections to achieve remarkable results on benchmarking datasets, a concrete explanation for the successful operation and improved generalization capability of these DNNs is surprisingly still lacking. In this paper, we focus on investigating the role of skip connections for training very deep DNNs. Our exposition directly provides interesting insights and new interpretations to the following important questions (i) why model optimization is easier (ii) why model generalization is better. Theoretical results reveal that skip connections allow DNNs to circumnavigate the singularity of latent representations that translate to optimization and generalization problems, which plague models without skip connections referred to as PlainNets. For substantiating our analysis, our investigation puts into context some of the most successful skip-connection based DNNs, which include residual networks (ResNets) and residual network with aggregated features (ResNeXt) in relation to PlainNets. Experimental evaluations of these models support the theoretical analysis.

Keywords: Deep neural network, residual learning, skip connection, optimization

1. Introduction

DEEP neural networks (DNNs) have become an indispensable tool for numerous learning tasks. Unprecedented applications and results on different difficult computer vision tasks have been reported using DNNs with several layers of features representations [1, 2, 3]. Lately, the computer vision community has started to theoretically investigate many techniques employed for various tasks; examples include DNN capacity [4], batch normalization [5] and variational auto encoders [6]. It is not enough to just construct 'model A' that outperforms 'model B', without a concrete explanation of why it is so. A fascinating empirical observation in recent times is the

*I am corresponding author

Email addresses: oyebade.oyedotun@uni.lu (Oyebade K. Oyedotun), kassem.alIsmaeil@uni.lu (Kassem Al Ismaeil), djamila.aouada@uni.lu (Djamila Aouada)

URL: https://wwwfr.uni.lu/snt/ (Oyebade K. Oyedotun)

positive correlation between the depth of DNNs and their generalization performance [7, 8]. In the past, DNN models mostly employed 2 to 4 hidden layers [9, 10], and did not use skip connections, hence referred to as *PlainNets*. However, there has been consistent increase in the number of model layers over time, e.g., Network-in-Network (NiN) [11] with 5 layers, AlexNet [12] with 8 layers, VGG-13 [13], and VGG-16 [13] and VGG-19 [13] with 13, 16 and 19 layers, respectively. Concurrently, there are theoretical works that study the benefit of model depth for learning complex target functions [14, 15, 16].
It is known that extending PlainNets' depths beyond some certain number of layers results in optimization problems; that is, the training set cannot be successfully learned [17, 18]. Subsequently, the difficulties of training such DNNs are alleviated by using skip connections of identity mappings [17, 19]; the works [17, 19] referred to the proposed DNN with skip connections as a residual network (*ResNet*).
Following the success of ResNets, the use of very complicated forms of skip connections can be seen in DNNs such as FractalNet [20], ResNeXt [21], PolyNet [22], DenseNet [23] and Inception-ResNet [24] where the summation or concatenation (or both) of previous layer representations with the current one is employed. In this paper, we theoretically and experimentally investigate the role of skip connections for training very deep DNNs. Specifically, we provide new interpretations to the role of skip connections in: 1)simplifying model optimizations, and 2)in improving model generalization. However, for the sake of theoretical analysis, we focus in this paper on DNNs that use skip connections of identity mappings and the summation of previous layers' representations with the current layer. Particularly, we consider two popular and very successful models, ResNet [17] and residual network with aggregated features (ResNeXt) [21]. Our contributions in this paper are as follows:

1. Theoretical analyses of the optimization characteristics of very deep DNNs with skip connections is provided where:
    - Training problems beyond saturating activation functions are identified using the power iteration method [25].
    - Generalization capabilities are explained based on the stability of learned parameters (i.e. solutions) for small changes in input or latent data.
2. Extensive experiments are performed to validate the theoretical analysis results using benchmarking datasets which include, MNIST, CIFAR-10, CIFAR-100 and ImageNet.

The remainder of this paper is organized as follows. In Section 2, we give an overview of the related work. The preliminaries on DNN with skip connections are provided in Section 3. In Section 4, the proposed theoretical analysis of the role of skip connections for different DNN models is presented. We further discuss the impact of skip connections on generalization in Section 5. Afterwards, the experiments and discussions are given in Section 6. Section 7 summarizes the paper as conclusion.

2. Related work

Several works have reported improved results on various tasks using DNNs that are deeper than previous state-of-the-art DNNs [26, 27, 28]. Furthermore, skip connections employed in top-down modulation framework was shown to improve the task of object detection in the work [29]. Consequently, emerging works naturally gravitate towards learning deeper DNNs, especially on hard tasks [30]. However, optimizing PlainNets beyond a certain depth is usually not successful [17, 18]. The work [31] noted that initialization schemes [32, 33] and batch normalization [34] do not resolve optimization problems beyond some certain model depth.

It has been shown that the optimization problems encountered in PlainNets can be circumvented by using skip connections [17, 18]. Some popular DNNs that have been successfully trained using this approach include ResNet [17], ResNeXt [21], highway network (HwNet) [18], densely connected network (DenseNet) [23], resnet of resnet (RoR) [35], dual path network (DPN) [36], PolyNet [22] and Inception-ResNet [24]. Although, the aforementioned models are very successful for learning different tasks, their operation has remained considerably elusive. Namely, *theoretical explanations* for how DNNs with skip connections manage to avoid optimization problems, and on top of that, generalize even better than PlainNets are still unclear.

In [37], optimization problems associated with gradients shattering are studied. Using carefully crafted experiments and auto correlation function values, it is observed that the gradients of shallow PlainNets are assimilated to brown noise. For deep PlainNets, the gradients are noted to be disorderly and resemble white noise. Interestingly, the gradients of ResNets are found to be well-behaved; Resnet gradients are considerably resistant; their structure is between white and brown noise. They posited that optimizing DNNs with gradients that resemble white noise is difficult; the closer gradients are to white noise, the more difficult optimization becomes.

In [38], it is observed that ResNets act like ensemble of shallow neural networks models; that is, their true depths are lesser than their architectural depth. In order to reveal this fascinating attribute of ResNets, the work in [38] provided and explored an unrolled view of ResNets. The work [38] further posited that paths through ResNets vary in length, and have small dependence on one another. It is concluded that ResNets avoid optimization problems by leveraging shorter paths in the models. In [39], an unrolled iterative estimation concept of features at different stages was proposed. The work argues that a collection of layers iteratively refines the approximations of similar features rather than computing wholely new features. New features are computed at other stages. Furthermore, the work in [40], using information bottleneck theory, related the encoding length of hidden layer representations to the regularization characteristics of DNNs. Our work differs from the aforementioned related works [37, 38, 39] in the approach of investigation and interpretations of model operation. In addition, [37] provides no explanation for improved generalization observed in the ResNet.

3. Background and preliminaries

3.1. Background: importance of studying linear models

For the ease of formalization, the linear activation function is assumed in this paper, as it is typical in the literature [41, 42, 43, 44, 45]. This is a very common assumption amongst other simplifications that make DNNs amenable to theoretical study; the convergence analysis of gradient descent

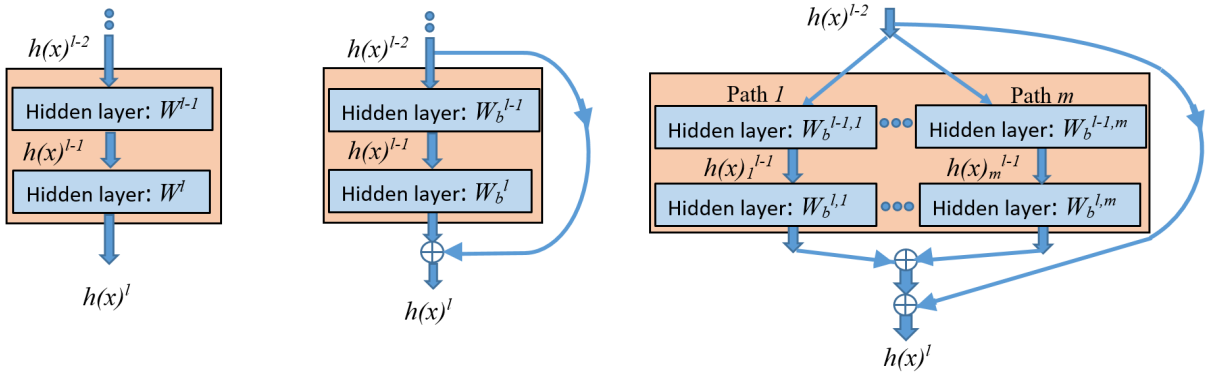

Figure 1: Models with skip connections considered in this paper. $\oplus$ denotes addition operation. Left: PlainNet block. Middle: ResNet block. Right: ResNeXt block. Full DNNs are constructed by stacking several blocks

on linear PlainNets is carried out in [46]. Furthermore, [44] discusses the relevance of studying linear DNNs, and posits that results are often relatable to DNNs with non-linear activation functions given that the loss function of a DNN (i.e., having two or more layers) with linear activation function is still non-convex with respect to the parameters space, as in DNNs with non-linear activation functions. Nevertheless, our experimental results use both non-linear and linear activation functions to show clear agreement with our theoretical analyses that assume linear activation function.

3.2. Preliminaries: PlainNet, ResNet and ResNeXt

In this section, the different forms of skip-connections based transformations for ResNet and ResNeXt are introduced as preliminaries. However, for a holistic investigation, we start with the PlainNet. Given model input data, $\boldsymbol{x}$, we use '$\boldsymbol{h}(\boldsymbol{x})$' to refer to the hidden representations.

(a) Plain network (PlainNet)

This category of DNNs represents the classical-vanilla DNNs that do not use skip connections of any form; see Figure 1. Each layer is connected via only one path to the the succeeding layer. Considering the hidden representation at layer l-2, $\boldsymbol{h}(\boldsymbol{x})^{l-2} \in \mathbb{R}^n$ (where $\boldsymbol{x} \in \mathbb{R}^n$ is the input to the DNN), for the two consecutive layer weights, W^l and W^{l-1}, in Figure 1, we can write the learned transformation as

$$\boldsymbol{h}(\boldsymbol{x})^l = \boldsymbol{W}^l \boldsymbol{W}^{l-1} \boldsymbol{h}(\boldsymbol{x})^{l-2}, \tag{1}$$

where $W^l, W^{l-1} \in \mathbb{R}^{n \times n}$ are random matrices.

(b) Residual network (ResNet)

Residual network (ResNet) [17] shown in Figure 1 essentially relies on skip connections of identity mappings that connect every residual block (containing 2 or 3 weight layers) to the previous one. For the ResNet block, we have

$$\boldsymbol{h}(\boldsymbol{x})^l = \boldsymbol{W}_b^l \boldsymbol{W}_b^{l-1} \boldsymbol{h}(\boldsymbol{x})^{l-2} + \boldsymbol{h}(\boldsymbol{x})^{l-2}, \tag{2}$$

where $\boldsymbol{W}_b^l, \boldsymbol{W}_b^{l-1} \in \mathbb{R}^{n \times n}$ are random matrices, and b indexes weights in a ResNet block. Furthermore, given the identity matrix $\mathbf{I}$, factorizing (2) results in

$$\boldsymbol{h}(\boldsymbol{x})^l = \left(\boldsymbol{W}_b^l \boldsymbol{W}_b^{l-1} + \mathbf{I}\right) \boldsymbol{h}(\boldsymbol{x})^{l-2}. \tag{3}$$

(c) Residual network with features aggregation (ResNeXt)
Residual network with features aggregation (ResNeXt) [21] relies on aggregating by summing features learned via m paths through every ResNeXt block as shown in Figure 1. For $1 \leq k \leq m$, the output of the ResNeXt block is

$$\boldsymbol{h}(\boldsymbol{x})^l = \sum_{k=1}^{m} \boldsymbol{W}_b^{l,k} \boldsymbol{W}_b^{l-1,k} \boldsymbol{h}(\boldsymbol{x})^{l-2} + \boldsymbol{h}(\boldsymbol{x})^{l-2}, \tag{4}$$

where $\boldsymbol{W}_b^{l,k}, \boldsymbol{W}_b^{l-1,k} \in \mathbb{R}^{n \times n}$ are random matrices, and b indexes weights in a ResNeXt block. Again, (4) can be simply factorized as

$$\boldsymbol{h}(\boldsymbol{x})^l = \left(\sum_{k=1}^{m} \boldsymbol{W}_b^{l,k} \boldsymbol{W}_b^{l-1,k} + \mathbf{I} \right) \boldsymbol{h}(\boldsymbol{x})^{l-2}. \tag{5}$$

4. Theoretical analysis of the role of skip connections for optimization

The central idea of this section is to investigate and analyse the role of skip connections presented in Section 3 in avoiding information loss while training a very deep DNN. Our analysis builds on the following assumption, propositions, corollary, definition and theorems.

Assumption 1. *For an L-layer DNN, all layer weight matrices, $\boldsymbol{W}^l \in \mathbb{R}^{n \times n}$, for $1 \leq l \leq L$, are assumed to be jointly diagonizable; implying the same eigenvectors $\{\boldsymbol{v}_i\}_{i=1}^n$, forming a basis in $\mathbb{R}^n$, and their corresponding eigenvalues $\{\lambda_i^l\}_{i=1}^n$.*

We note that a similar assumption can be found in the work [47].
Our analysis starts first on the forward-pass and then on the backpropagation in relation to optimization conditions.

Proposition 1. *For a matrix $W \in \mathbb{R}^{n \times n}$ whose column vectors, $\boldsymbol{w}_i$, are randomly drawn from a uniform distribution $U[-r, r]$ or a Gaussian distribution $\mathcal{N}(\mu, \sigma^2)$, the probability that W is non-singular, $P(\boldsymbol{w_i} \notin \boldsymbol{W}_{-i}^{span})$, is*

$$P(\boldsymbol{w}_i \notin \boldsymbol{W}_{-i}^{span}) = 1 \ : 1 \leq i \leq n \tag{6}$$

Proof. See Section A1 in the appendix. □

Corollary 1. *Considering that DNN parameters, $\boldsymbol{W}^l : 1 \leq l \leq L$, are randomly initialized [32, 33], W^l is therefore guaranteed to be a non-singular matrix at the start of training.*

Proof. This follows directly from Proposition 1. □

Definition 1. *Given $\{\boldsymbol{v_i}\}_{i=1}^n$ is a set of eigenvectors (i.e. orthogonal vectors) as in Assumption 1, any $\boldsymbol{x} \in \mathbb{R}^n$ can be expressed as*

$$\boldsymbol{x} = \sum_{i=1}^{n} \alpha_i \boldsymbol{v_i}, \ \ \alpha_i \in \mathbb{R}. \tag{7}$$

Definition 2. *A batch of N hidden representations, $\boldsymbol{H}(\boldsymbol{x}) \in \mathbb{R}^{n \times N}$, is composed of N individual data points, $\boldsymbol{h}(\boldsymbol{x})_i \in \mathbb{R}^n$; that is, $\boldsymbol{H}(\boldsymbol{x}) = [\boldsymbol{h}(\boldsymbol{x})_1, \cdots, \boldsymbol{h}(\boldsymbol{x})_i, \cdots, \boldsymbol{h}(\boldsymbol{x})_N]$.*

Proposition 2. *Given a non-invertible matrix $\boldsymbol{M} \in \mathbb{R}^{n \times n}$, and vector $\boldsymbol{v} \in \mathbb{R}^n$, then the transformation $\boldsymbol{s} = \boldsymbol{M}\boldsymbol{v}$ incurs information loss.*

Proof. First note that a non-invertible matrix is singular, has a determinant of zero, and its transformation collapses space. Consequently, $\boldsymbol{v}$ cannot be recovered with certainty by $\boldsymbol{M}^{\dagger}\boldsymbol{s}$, where $\dagger$ denotes the pseudo-inverse of a matrix, since the mapping $\boldsymbol{v} \mapsto \boldsymbol{s}$ is non-injective. □

It is critical to note that *singularity*, where at least one singular value of M becomes zero is not necessary for optimization problems; *near-singularity*, where at least one singular value is extremely small is sufficient. This is directly reflected in the condition number κ of $\boldsymbol{M}$ defined as

$$\kappa(\boldsymbol{M}) = \sigma_{max}(\boldsymbol{M})/\sigma_{min}(\boldsymbol{M}), \tag{8}$$

where $\sigma_{max}(\boldsymbol{M})$ and $\sigma_{min}(\boldsymbol{M})$ are the maximum and minimum singular values of $\boldsymbol{M}$, respectively. Problems with $\kappa(\boldsymbol{M}) \gg 1$ are commonly said to be ill-posed.

4.1. Forward-pass: loss of basis in hidden representations

(a) Plain network (PlainNet)

For the PlainNet in Section 3 (a), we state Theorem 1.

Theorem 1. *Given an input $\boldsymbol{x} \in \mathbb{R}^n$ to a linear L-layer PlainNet parameterized as $\{\boldsymbol{W}^{\boldsymbol{l}}\}_{l=1}^{L} \in \mathbb{R}^{n \times n}$, the hidden layer output, $\boldsymbol{h}(\boldsymbol{x})^L : L \gg 1$, is*

$$\begin{cases} \boldsymbol{h}(\boldsymbol{x})^L \approx \prod_{l=1}^{L} \lambda_1^l \left(\alpha_1 \boldsymbol{v}_1\right), \\ \textit{s.t.} \quad \left|\lambda_1^l\right| > \left|\lambda_i^l\right| \quad : \; 2 \leq i \leq n, \end{cases} \tag{9}$$

where λ_i^l are the first components of the eigenvalues. The basis $\{\boldsymbol{v}_i\}$ is the eigenvectors of W^l, with $l = 1, \cdots, L$, and the scalar α_1 is the first coordinate in this space.

Proof. The power iteration method [25] is modified (since different weights are applied at the different layers of the PlainNet) for analysing the impact of successive layer weights transformations on $\boldsymbol{h}(\boldsymbol{x})^L$ as follows.

Using Assumption 1 for layer l, we can write

$$\boldsymbol{W}^l \boldsymbol{v}_i = \lambda_i^l \boldsymbol{v}_i. \tag{10}$$

For the final PlainNet output at layer L, we can write

$$\boldsymbol{h}(\boldsymbol{x})^L = \prod_{l=1}^{L} \boldsymbol{W}^l \boldsymbol{x}. \tag{11}$$

Putting (7) and (10) into (11) and considering Assumption 1, we have (see Section A2 in the appendix)

$$\boldsymbol{h}(\boldsymbol{x})^L = \sum_{i=1}^{n} \alpha_i \boldsymbol{v}_i \prod_{l=1}^{L} \lambda_i^l, \tag{12}$$

which can be rewritten as

$$\boldsymbol{h}(\boldsymbol{x})^L = \left(\alpha_1 \boldsymbol{v}_1 \prod_{l=1}^{L} \lambda_1^l + \sum_{i=2}^{n} \alpha_i \boldsymbol{v}_i \prod_{l=1}^{L} \lambda_i^l \right). \tag{13}$$

Furthermore, (13) can be factorized so that

$$\boldsymbol{h}(\boldsymbol{x})^L = \prod_{l=1}^{L} \lambda_1^l \left(\alpha_1 \boldsymbol{v}_1 + \sum_{i=2}^{n} \alpha_i \boldsymbol{v}_i \prod_{l=1}^{L} \left(\lambda_i/\lambda_1\right)^l \right). \tag{14}$$

Finally, (9) is obtained from (14) given that $\prod_{l=1}^{L} \left(\lambda_i/\lambda_1\right)^l \longrightarrow 0$ for $L \gg 1$ and $\left|\lambda_1^l\right| > \left|\lambda_i^l\right| \ : \ 2 \leq i \leq n$, as expected for eigenvalues. □

Ultimately, note that a similar observation $\prod_{l=1}^{L} \left(\lambda_i/\lambda_1\right)^l \longrightarrow 0$ for $L \gg 1$ is used for concluding the following theorems.

Remark 1. *For an input $\boldsymbol{x}$, only one basis vector, $\boldsymbol{v_1}$, contributes to the computation of $\boldsymbol{h}(\boldsymbol{x})^L$ for $L \gg 1$, as a result of repeated multiplication by $\{\boldsymbol{W}^l\}_{l=1}^{L}$. As such, $\boldsymbol{h}(\boldsymbol{x})^L$ for the PlainNet exhibits significant information loss. Considering Definition 2 and Theorem 1, the columns of $\boldsymbol{H}(\boldsymbol{x})^L$ are colinear; hence, $\boldsymbol{H}(\boldsymbol{x})^L$ is singular.*

Going by Remark 1, we observe that [48] used deep Gaussian process to arrive at a similar result that the representational capacity of DNNs collapses to one degree of freedom as $L \longrightarrow \infty$. Furthermore, it can be concluded from Theorem 2 that the information loss of hidden representations is not associated with the popular unit saturation problem [32], as Theorem 2 uses linear units.

(b) Residual network (ResNet)

For analyzing features conditioning of ResNet given in Section 3 (b), we rely on the following lemma and axiom.

Lemma 1. *For a matrix $\boldsymbol{W} \in \mathbb{R}^{n \times n}$ with eigenvector $\boldsymbol{v} \in \mathbb{R}^n$ and eigenvalue λ such that $\boldsymbol{W}\boldsymbol{v} = \lambda\boldsymbol{v}$, it can be shown that $(\boldsymbol{W} + c\mathbf{I})\boldsymbol{v} = (\lambda + c)\boldsymbol{v}$ for a scalar c.*

Proof. $(\boldsymbol{W} + c\mathbf{I})\boldsymbol{v} = \boldsymbol{W}\boldsymbol{v} + c\boldsymbol{v} = \lambda\boldsymbol{v} + c\boldsymbol{v} = (\lambda + c)\boldsymbol{v}$. □

Axiom 1. *For sequential products of a sum of variable and constant, we can write $\prod_{l=1}^{L}(\lambda^l + 1) = 1 + \sum_{l=1}^{L} \lambda^l + ... + \prod_{l=1}^{L} \lambda^l$, where the middle terms are inconsequential for stating the next theorem.*

In addition, results of transformations $\boldsymbol{W}_b^l \boldsymbol{W}_b^{l-1}$ in a residual block in (3) is lumped as a single transformation

$$\boldsymbol{W}^l = \boldsymbol{W}_b^l \boldsymbol{W}_b^{l-1}. \tag{15}$$

Theorem 2. *Given an input $\boldsymbol{x} \in \mathbb{R}^n$ to a linear L-layer ResNet parameterized as $\{\boldsymbol{W}^l\}_{l=1}^L \in \mathbb{R}^{n\times n}$, the hidden layer output, $\boldsymbol{h}(\boldsymbol{x})^L : L \gg 1$, is*

$$\begin{cases} \boldsymbol{h}(\boldsymbol{x})^L \approx \sum_{i=1}^{n} \alpha_i \boldsymbol{v}_i + \alpha_1 \boldsymbol{v}_1 \left(\sum_{l=1}^{L} \lambda_1^l + \cdots + \prod_{l=1}^{L} \lambda_1^l \right) \\ \qquad\qquad\qquad\qquad \textit{s.t. } \left|\lambda_1^l\right| > \left|\lambda_i^l\right| \quad : \ 2 \leq i \leq n. \end{cases} \tag{16}$$

Proof. Again, the modified power iteration method is employed. Using (3) and (15), we can write for the ResNet

$$\boldsymbol{h}(\boldsymbol{x})^{\boldsymbol{L}} = \prod_{l=1}^{L} (\boldsymbol{W}^l + \mathbf{I})\boldsymbol{x}. \tag{17}$$

Applying Lemma 1 with $c = 1$ gives $(\boldsymbol{W}^l + \mathbf{I})\boldsymbol{v}_i = (\lambda_i^l + 1)\boldsymbol{v}_i$; putting (7) into (17) and considering Assumption 1 gives

$$\boldsymbol{h}(\boldsymbol{x})^L = \sum_{i=1}^{n} \alpha_i \boldsymbol{v}_i \prod_{l=1}^{L} (\lambda_i^l + 1). \tag{18}$$

Using Axiom 1 for (18), we can write

$$\boldsymbol{h}(\boldsymbol{x})^L = \sum_{i=1}^{n} \alpha_i \boldsymbol{v}_i \left(1 + \sum_{l=1}^{L} \lambda_i^l + \cdots + \prod_{l=1}^{L} \lambda_i^l \right). \tag{19}$$

Furthermore, (19) can be expanded as

$$\begin{aligned} \boldsymbol{h}(\boldsymbol{x})^L = & \sum_{i=1}^{n} \alpha_i \boldsymbol{v}_i + \alpha_1 \boldsymbol{v}_1 \left(\sum_{l=1}^{L} \lambda_1^l + ... + \prod_{l=1}^{L} \lambda_1^l \right) + \\ & \sum_{i=2}^{n} \alpha_i \boldsymbol{v}_i \left(\sum_{l=1}^{L} \lambda_i^l + ... + \prod_{l=1}^{L} \lambda_i^l \right). \end{aligned} \tag{20}$$

The factorization of (20) yields

$$\begin{aligned} \boldsymbol{h}(\boldsymbol{x})^L = & \sum_{i=1}^{n} \alpha_i \boldsymbol{v}_i + \left(\sum_{l=1}^{L} \lambda_1^l + ... + \prod_{l=1}^{L} \lambda_1^l \right) \Bigg\{ \alpha_1 \boldsymbol{v}_1 + \\ & \sum_{i=2}^{n} \alpha_i \boldsymbol{v}_i \left(\frac{\sum_{l=1}^{L} \lambda_i^l + ... + \prod_{l=1}^{L} \lambda_i^l}{\sum_{l=1}^{L} \lambda_1^l + ... + \prod_{l=1}^{L} \lambda_1^l} \right) \Bigg\}. \end{aligned} \tag{21}$$

Lastly, applying $L \gg 1$ and the condition $\left|\lambda_1^l\right| > \left|\lambda_i^l\right| \ : \ 2 \leq i \leq n$, as expected for eigenvalues to (21) completes the proof of Theorem 2. □

Remark 2. *Given the input $\boldsymbol{x}$, all basis vectors $\{\boldsymbol{v}_i\}_{i=1}^n$, contribute to the computation of $\boldsymbol{h}(\boldsymbol{x})^L$ for $L \gg 1$. As such, $\boldsymbol{h}(\boldsymbol{x})^L$ for the ResNet retain full information. From Definition 2 and Theorem 2, the columns of $\boldsymbol{H}(\boldsymbol{x})^L$ are distinct, and therefore $\boldsymbol{H}(\boldsymbol{x})^L$ is non-singular.*

(c) ResNeXt

The following ResNeXt analysis is based on Section 3 (c)

Theorem 3. ***Theorem 3*** *Given an input $\boldsymbol{x} \in \mathbb{R}^n$ to a linear L-layer ResNeXt parameterized as $\{\boldsymbol{W}^{l,k}\}_{l=1}^{L}$, the hidden layer output, $\boldsymbol{h}(\boldsymbol{x})^L : L \gg 1$, is*

$$\begin{cases} \boldsymbol{h}(\boldsymbol{x})^L \approx \sum_{i=1}^{n} \alpha_i \boldsymbol{v}_i + \alpha_1 \boldsymbol{v}_1 \left(\sum_{l=1}^{L} \sum_{j=1}^{m} \lambda_1^{l,m} + ... + \prod_{l=1}^{L} \sum_{j=1}^{m} \lambda_1^{l,m} \right) \\ \qquad s.t. \ \left|\lambda_1^l\right| > \left|\lambda_i^l\right| \quad : \ 2 \leq i \leq n. \end{cases} \tag{22}$$

Proof. See Section A3 in the appendix. □

Remark 3. *Given the input $\boldsymbol{x}$, all basis vectors $\{\boldsymbol{v}_i\}_{i=1}^{n}$, contribute to the computation of $\boldsymbol{h}(\boldsymbol{x})^L$ for $L \gg 1$. Similar to ResNet, $\boldsymbol{h}(\boldsymbol{x})^L$ for the ResNeXt retain full information. Again, employing Definition 2 and Theorem 3 shows that the columns of $\boldsymbol{H}(\boldsymbol{x})^L$ are distinct, and thus $\boldsymbol{H}(\boldsymbol{x})^L$ is non-singular.*

4.2. Backpropagation: error gradients singularity

It is well known that the trainability of DNNs is subject to *well-conditioned* error gradients; otherwise, optimization is unlikely to converge. For the sake of simplicity, we use the following lemmas in the subsequent parts.

Lemma 2. *Let $\boldsymbol{X} \in \mathbb{R}^{m \times n}$ and $\boldsymbol{V} \in \mathbb{R}^{n \times p}$ be matrices. If $\boldsymbol{V}$ is singular, then their product, $\boldsymbol{Y} = \boldsymbol{X}\boldsymbol{V}$, is also singular.*

Proof. Some row(s) or column(s) of $\boldsymbol{V}$ are linearly correlated. As such, $\exists$ a vector $\boldsymbol{u} \in \mathbb{R}^n$ with $\| \boldsymbol{u} \|_2 \neq 0$: $\boldsymbol{V}\boldsymbol{u} = 0$. Hence, $\boldsymbol{X}\boldsymbol{V}\boldsymbol{u} = 0$, and thus $\boldsymbol{X}\boldsymbol{V}$ is singular. □

Lemma 3. *Let $\boldsymbol{X} \in \mathbb{R}^{m \times n}$ and $\boldsymbol{V} \in \mathbb{R}^{n \times p}$ be two matrices. If $\boldsymbol{X}$ and $\boldsymbol{V}$ are non-singular, then their product, $\boldsymbol{Y} = \boldsymbol{X}\boldsymbol{V}$, is also non-singular.*

Proof. All row(s) or column(s) of both $\boldsymbol{X}$ and $\boldsymbol{V}$ are linearly uncorrelated. As such, $\nexists$ a vector $\boldsymbol{u} \in \mathbb{R}^n$ with $\| \boldsymbol{u} \|_2 \neq 0$ such that $\boldsymbol{X}\boldsymbol{u} = 0$ or $\boldsymbol{V}\boldsymbol{u} = 0$. Consequently, $\boldsymbol{X}\boldsymbol{V}\boldsymbol{u} \neq 0$, and thus $\boldsymbol{X}\boldsymbol{V}$ is non-singular. □

(a) Plain network (PlainNet)

Definition 3. *Given the outputs of units in a hypothetical layer l is $\boldsymbol{H}(\boldsymbol{x})^l$, the local error gradient of units, $\boldsymbol{\Delta}^l$, that is connected to units in layer $(l+1)$ is of the form*[1]

$$\boldsymbol{\Delta}^l = \boldsymbol{H}(\boldsymbol{x})^l \left(\boldsymbol{W}^{l+1} \boldsymbol{\Delta}^{l+1} \right). \tag{23}$$

[1]Given that the linear activation function is employed

For training cost, E, the weight update for $\boldsymbol{W}^l$ at iteration t using the learning rate η is

$$\boldsymbol{\Delta W}^l(t) = -\eta \frac{\partial E}{\partial \boldsymbol{W}^l} = \eta \Delta^l \boldsymbol{H}(\boldsymbol{x})^{l-1}, \tag{24}$$

$$\text{with} \quad \boldsymbol{W}^l(t) = \boldsymbol{W}^l(t-1) + \boldsymbol{\Delta W}^l(t). \tag{25}$$

Given that DNN parameters by convention are randomly initialized and are generally quite far from the solution, large weight updates are made at the start of training. As such, if the total number of iterations is T, we can write

$$\boldsymbol{W}^l(t) \approx \boldsymbol{\Delta W}^l(t) \quad : 1 \leq t \ll T. \tag{26}$$

Remark 4. *Using Remark 1 that $\boldsymbol{H}(\boldsymbol{x})^l$ exhibits singularity and Lemma 2, it can be concluded from* (23) *that the error gradient estimate, $\boldsymbol{\Delta}^l$, is singular. Subsequently, via a similar argument, $\boldsymbol{\Delta W}^l$ in* (24) *is singular, and so is $\boldsymbol{W}^l$ in* (26)*. Thus, optimization exhibits numerical instability [49, 50].*

From Remark 4, DNN computational results lose precision, and errors can accrue to cause significant erratic weights updates, and therefore non-convergence. This observation is interesting, as it directly corroborates the work in [37] that studied error gradients as white noise in very deep PlainNets. Our experimental results indeed confirm this.

(b) ResNet and ResNeXt

Remark 5. *Using Remark 2 that $\boldsymbol{H}(\boldsymbol{x})^l$ is non-singular and Corrollary 1 that $\boldsymbol{W}^{l+1}$ is non-singular, it follows from Lemma 3 and* (23) *that the error gradient, $\boldsymbol{\Delta}^l$, is non-singular. Likewise, from* (24) *and* (26)*, weights updates and weights are both non-singular and decent estimates for successful optimization.*

Remark 6. *Given that $\boldsymbol{H}(\boldsymbol{x})^l$ is non-singular from Remark 3 and $\boldsymbol{W}^{l+1}$ is non-singular from Corrollary 1, it follows from Lemma 3 and* (23) *that the error gradient, $\boldsymbol{\Delta}^l$, is also non-singular. Similar to ResNet, note the non-singularity of weight updates and weights in* (24) *and* (26)*, respectively, which contribute to a successful optimization.*

5. Skip connections impact generalization

We show in the following proposition that the condition of input data and hidden representations not only affect model optimization, but also reflects in the implicit regularization of learned parameters.

Proposition 3. *For a DNN with optimal solution $\boldsymbol{\theta}$, $\boldsymbol{\Delta H}(\boldsymbol{x})^l$ translates to a relative change in solution, $\Delta\boldsymbol{\theta}$, as follows*

$$\frac{\parallel \Delta\boldsymbol{\theta} \parallel}{\parallel \boldsymbol{\theta} \parallel} \leq \kappa(\boldsymbol{H}(\boldsymbol{x})^l) \frac{\parallel \Delta \boldsymbol{H}(\boldsymbol{x})^l \parallel}{\parallel \boldsymbol{H}(\boldsymbol{x})^l \parallel} \quad : 0 \leq l \leq L, \tag{27}$$

where $\boldsymbol{H}(\boldsymbol{x}) \in \mathbb{R}^{n \times N}$ and $\boldsymbol{H}(\boldsymbol{x})^0 = [\boldsymbol{x}_1, ..., \boldsymbol{x}_N]$ for $l = 0$.

Model	Train accuracy	Test accuracy
PlainNet-110	11.20%	11.35%
ResNet-110	99.99%	99.57%
ResNeXt-110	99.99%	99.71%

Table 1: Results after training on the MNIST dataset

Model	Train accuracy	Test accuracy
PlainNet-110	10.26%	10.05%
ResNet-110	99.98%	93.29%
ResNeXt-110	99.98%	93.65%

Table 2: Results after training on the CIFAR-10 dataset

Proof. See Section A3 in the appendix. □

From Proposition 3, it is seen that the *fragility* of the solution θ depends on $\kappa(\boldsymbol{H}(\boldsymbol{x})^l)$. We show in the experiments that DNNs with skip connections operate with much smaller condition numbers than PlainNets. As such, a change $\Delta\boldsymbol{H}(\boldsymbol{x})^l$ (that can occur when we move from the training set to test set) translates to a smaller change in $\boldsymbol{\theta}$ for DNNs with skip connections than PlainNets. Hence, a more stable solution reflects in improved generalization.

6. Experiments

6.1. Datasets, settings and evaluations

For the experiments, MNIST [51], CIFAR-10 [52], CIFAR-100 [52] and ImageNet-2012 [53] datasets are used. All experiments except on MNIST dataset use standard data augmentation as in [54]. Very deep PlainNet, ResNet and ResNeXt with 110 and 200 layers are trained on MNIST, CIFAR-10 and CIFAR-100 datasets. For ImageNet-2012 dataset, only ResNet [17] and ResNeXt [21] models with 101-layers are trained. Mini-batch gradient descent, learning rate in the range [0.0001-0.1], momentum rate in the range [0.1-0.9], weight decay of 10^{-4}, batch size of 128 and a maximum of 200 epochs are used for training; all model parameters are initialized as random Gaussian tensors using [32]. All our experiments use the rectified linear units (ReLUs) to demonstrate the agreement between theoretical analysis and practical DNNs. Namely, we aim to observe for PlainNets, ResNet and ResNeXt, the following (i) evolution of units' activations with depth (ii) weights updates during training (iii) error gradients during training (iv) progressive near-singularity of model layer weights with depth, which is measured via condition number computed from singular values as in (8), and (v) information loss of hidden layer representations with depth via near-singularity (i.e. condition number). The singular values for model tensors are obtained using Higher Order Singular value decomposition (HOSVD) [55], which generalizes the well-known matrix SVD to n-dimensional arrays (i.e tensors).

6.2. Results and discussion

The experimental results of our investigation are discussed as follows. Tables 1, 2, 3 & 4 show the training and testing accuracies on the other three datasets. The unsuccessful training of the PlainNet is clearly reflected in the obtained training and testing accuracies. We note that

Model	Train accuracy	Test accuracy
PlainNet-101	3.72%	3.68%
ResNet-101	99.84%	72.27%
ResNeXt-101	99.99%	72.43%

Table 3: Results after training on CIFAR-100 dataset

Model	Train accuracy	Test accuracy
PlainNet-101	15.37%	14.94%
ResNet-101	88.56%	77.32%
ResNeXt-101	89.74%	78.48%

Table 4: Results (i.e. top-1 accuracy) after training on the ImageNet dataset

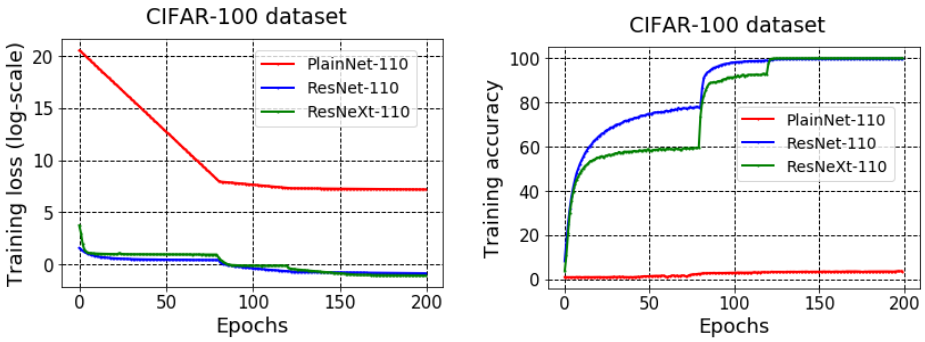

Figure 2: Training curves for models. Left: Training loss. Right: Training accuracy

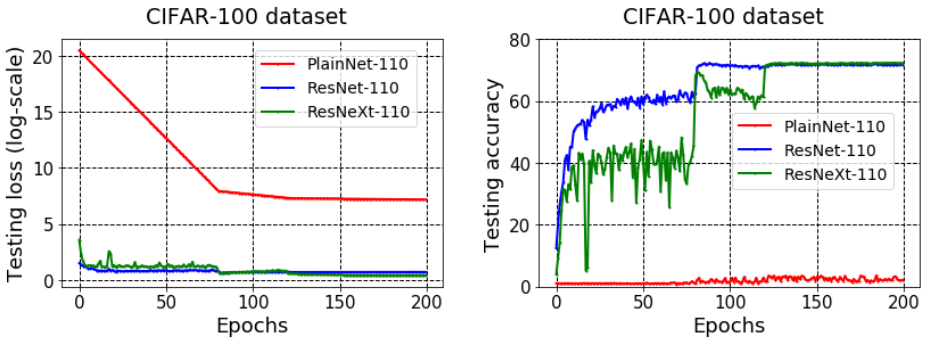

Figure 3: Testing curves for models. Left: Testing loss. Right: Testing accuracy

hyperparameters search did not yield trainable PlainNets. Figure 2 and Figure 3 show the training and testing curves, respectively, for the different models trained on CIFAR-100 dataset. Note that the training and testing losses are plotted to log-scale due to extremely high values for the PlainNet. The difficult of optimizing the PlainNet is evident in the extremely large training loss and poor training accuracy observed. Figure 4 shows the units' activations for PlainNet, ResNet and ResNeXt with 110 layers trained CIFAR-100 dataset. For observing the training conditions of the aforementioned DNNs, early and later layers are studied. Consequently, layers 1 and 108 are taken

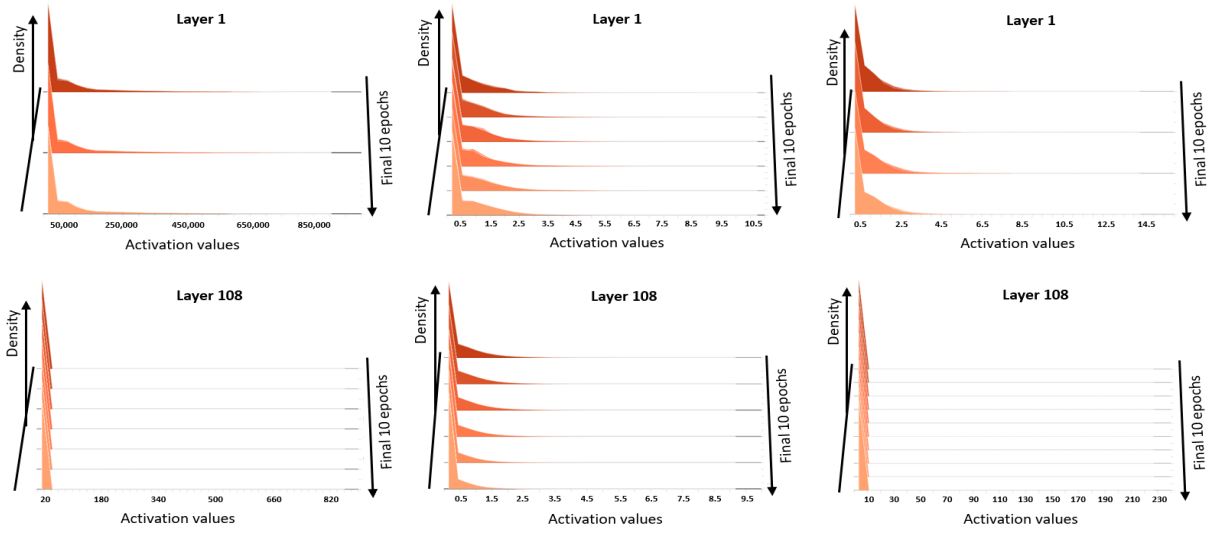

Figure 4: Units' activations for 110 layer models trained on the CIFAR-100 dataset. Left column: PlainNet. Middle column: ResNet. Right column: ResNeXt

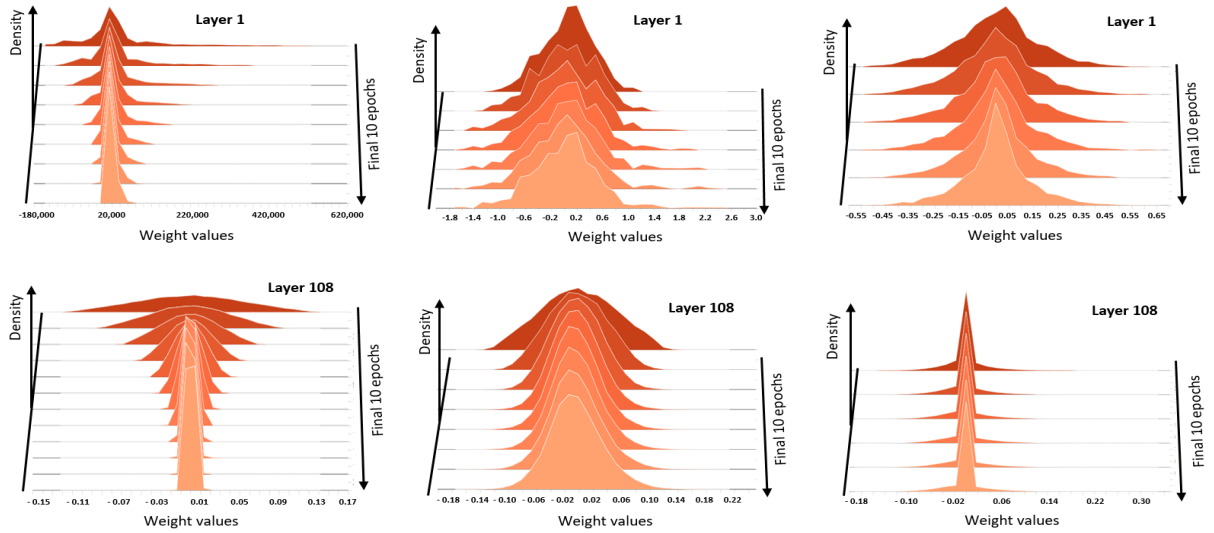

Figure 5: Units' weights for 110 layer models trained on the CIFAR-100 dataset. Left column: PlainNet. Middle column: ResNet. Right column: ResNeXt

for inspection. It is seen in the layer 1 of the PlainNet that most units have very large activations (i.e. up to $\approx 10^6$) that are problematic. For ResNet and ResNeXt models, activations in layer 1 are less than 6.5. At layer 108, all models operate with reasonable units' activitions values.
Figure 5 shows the progress of weights update for the PlainNet, ResNet and ResNeXt given in Figure 4. For layer 1, it is seen that the PlainNets have weight values in the problematic range $-180,000$ to $620,000$. Conversely, ResNet and ResNeXt weights are in the reasonable ranges of -1.8 to 2.6 and -0.55 to 0.65, respectively. At layer 108, the PlainNet's weight values progressively collaps to zero, while the ResNet and ResNeXt weights both have fairly consistent values

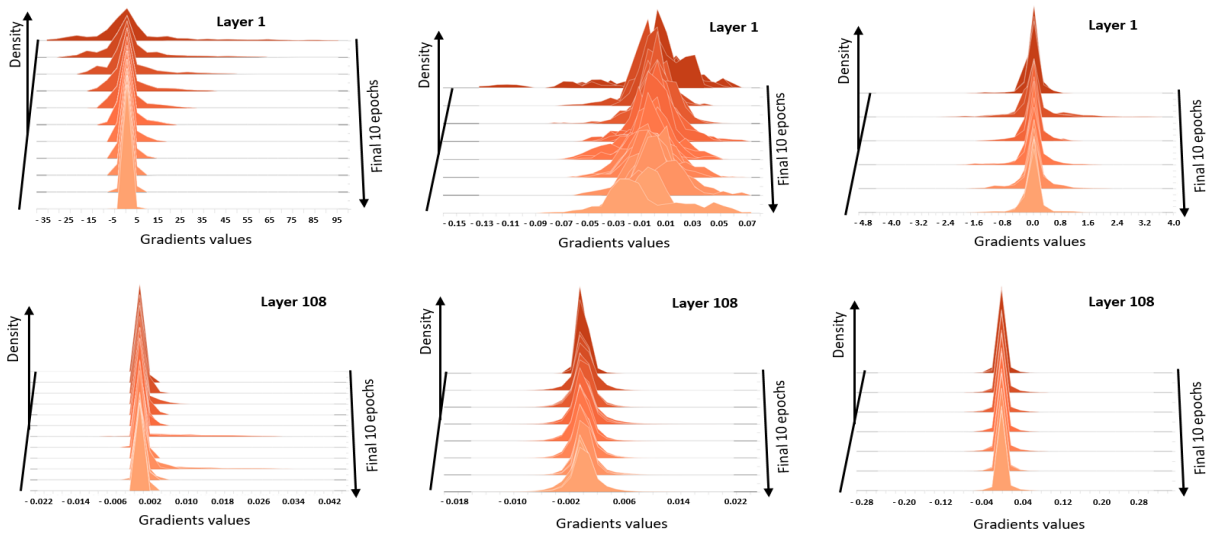

Figure 6: Units' gradients for 110 layer models trained on CIFAR-100 dataset. Left column: PlainNet. Middle column: ResNet. Right column: ResNeXt

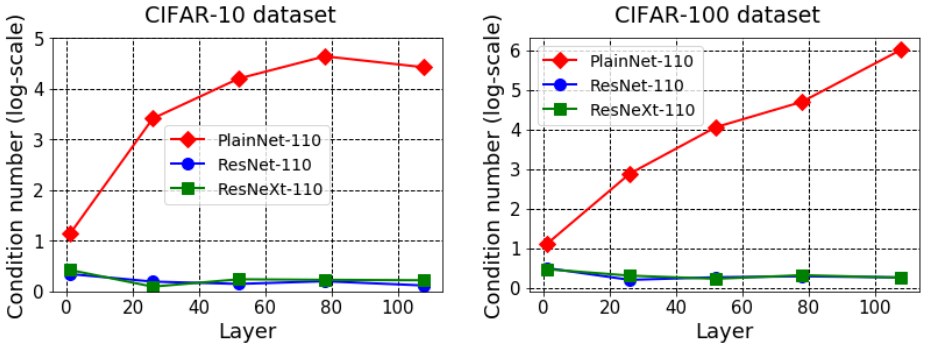

Figure 7: Model weights condition number. Left: CIFAR-10 dataset. Right: CIFAR-100 dataset

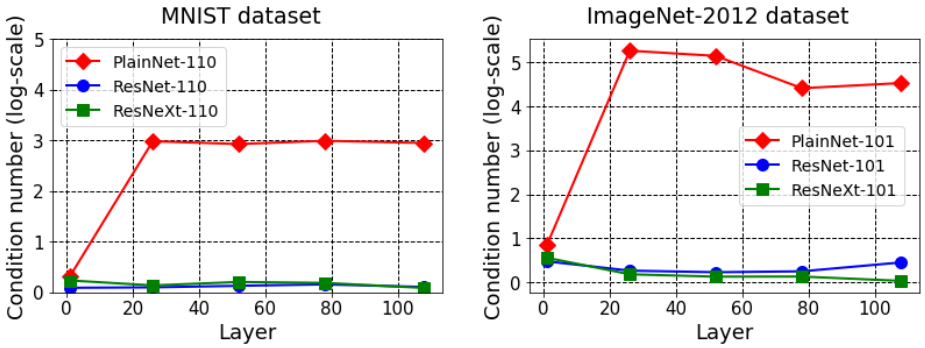

Figure 8: Model weights condition number. Left: MNIST dataset. Right: ImageNet dataset

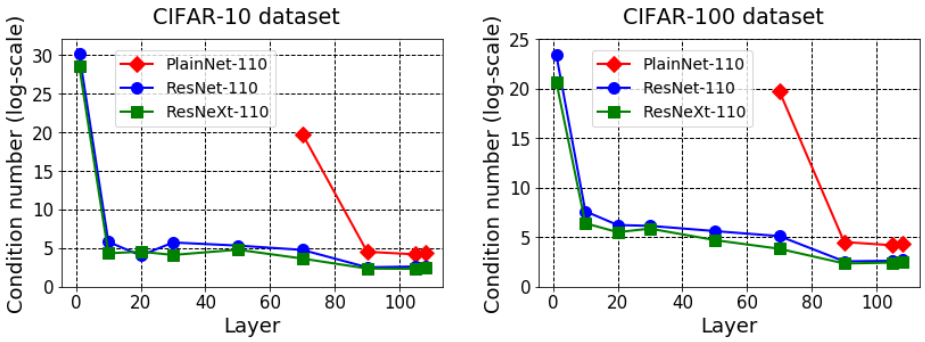

Figure 9: Model hidden layer condition number; layers with missing condition numbers have infinite values. Left: CIFAR-10 dataset. Right: CIFAR-100 dataset

in the reasonable range -0.18 to 0.30.

Figure 6 shows model gradients. Most gradients in layer 1 of the PlainNet are considerably large in the range -5 to 5; large gradients depict optimization problems. However, ResNet and ResNeXt have most gradients in the reasonable range -0.07 to 0.07 and -0.8 to 0.8, respectively. The gradients for the PlainNet, ResNet and ResNeXt at layer 108 are at reasonable values of -10^{-2} to 10^{-2}. Note that Figure 4, Figure 5 and Figure 6 are unnormalized to directly convey the severity of the the training problems.

Figure 7 and Figure 8 show the condition number of model weights. It can be seen that the condition number of the PlainNet weights grows with depth, indicating that the learned model is progressively ill-posed with depth increase. Interestingly, the condition numbers of ResNet and ResNeXt weights remain small with depth increase; this indicates good stability of learned parameters in line with proposition 3. Figure 9 shows the condition number of hidden layer activations for the trained PlainNet, ResNet and ResNeXt. As such, the PlainNets have infinite condition numbers from layer 1 to 70. Note that infinite condition number indicates perfect singularity, and thus the worst scenario of optimization difficulty. The early layers in the ResNet and ResNeXt have high condition numbers (but no perfect singularity) that rapidly reduces to allow successful optimization.

6.3. Results for deeper DNN models

In this section, to reinforce the results obtained in Section 6.2 with models having 110 and 101 layers, we report the results of additional experiments using models with 200 layers trained on the MNIST dataset. Specifically, we show that models, which are much deeper than those reported in Section 6.2 show similar training characteristics. The 200-layer models for the PlainNet, ResNet and ResNeXt are referred to as PlainNet-200, ResNet-200 and ResNeXt-200, respectively. We note that PlainNet-200 achieved training and testing accuracies of 11.23% and 8.92%, respectively; this shows poor optimization. Interestingly, ResNet-200 and ResNeXt-200 both reach training and testing accuracies of over 99%, showing successful optimization.

Figure 10 shows the condition numbers of the weights in the different layers in the models. It is seen that PlainNet-200 generally has higher condition numbers than the ResNet-200 and

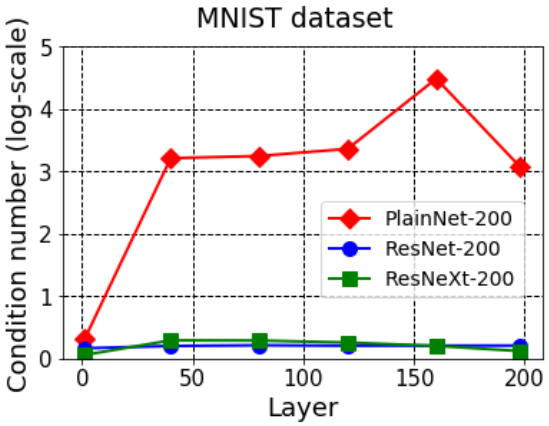

Figure 10: Model weights condition number

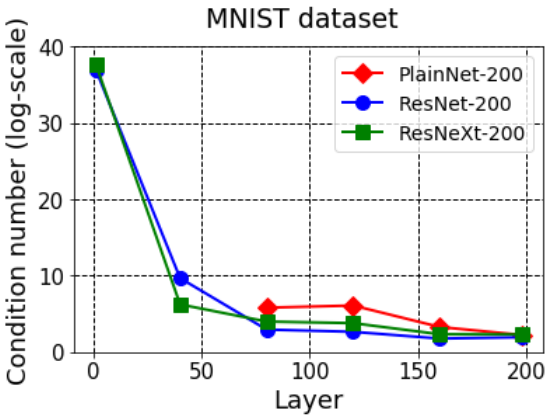

Figure 11: Model hidden layer condition number; layers with missing condition numbers have infinite values

ResNeXt-200. Furthermore, Figure 11 shows the condition numbers of the hidden representations in the different layers of the models. It is seen that the hidden representations of PlainNet-200 has infinite condition numbers up to the 80th layer. In contrast, the hidden representations of ResNet-200 and ResNeXt-200 both have finite condition numbers for all layers. Ultimately, these observations about the conditions of the weights and hidden representations are similar to that in Section 6.2 for models having 110 and 101 layers.

6.4. Results for linear models

In Sections 6.2 and 6.3, the reported experimental results use the rectified linear activation function (i.e. rectified linear units: ReLUs) for practicality reasons. That is, to show that our theoretical results in Section 4 and Section 5, which assumed the linear activation function are valid for the non-linear activation function. Herein, we perform additional experiments using the linear activation function for the different models trained on the MNIST dataset [51] to show that similar conclusions can be made. Specifically, we train the PlainNet, ResNet and ResNeXt with 164 layers using the linear activation function (i.e. with linear units). The condition numbers for the different

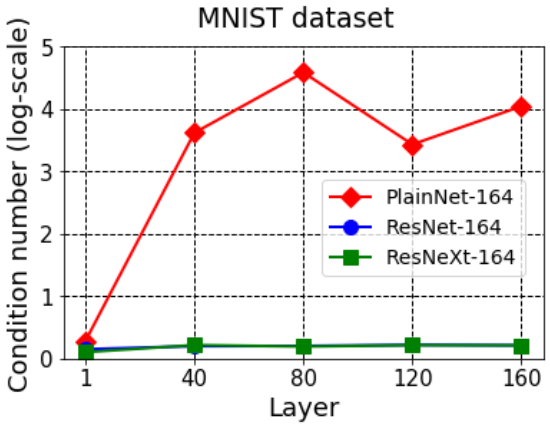

Figure 12: Condition number for layer weights in the models trained with the linear activation function

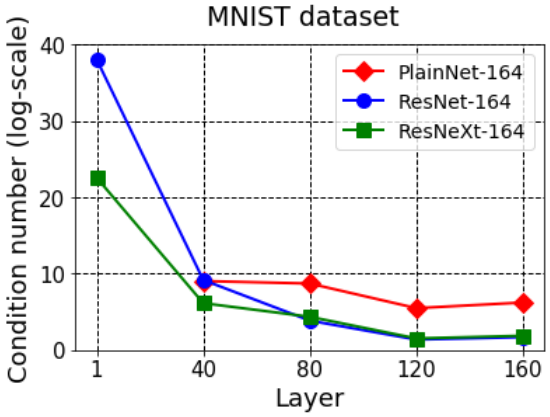

Figure 13: Condition number for hidden layer representations in the models trained with the linear activation function; layers with missing condition numbers have infinite values

layer weights in the trained models are shown in Figure 12. Similar to models with ReLUs, it is seen that the PlainNet layer weights have high condition numbers as compared to the ResNet and ResNeXt; compare Figure 12 with Figure 7 and Figure 8. Furthermore, Figure 13 shows the condition numbers for the different hidden layer representations in the trained models. Again, the PlainNet operates with high condition number as compared to the ResNet and ResNeXt. In fact, the hidden representation in the first layer of the PlainNet has an infinite condition number; this is the worst scenario for optimization problems to ensue. However, the ResNet and ResNeXt have finite condition numbers for the hidden representations in different the layers. Importantly, the condition of the hidden representations in the models with linear units is similar to the models with ReLUs; compare Figure 13 with Figure 9.

6.5. Skip connections in hindsight

(a) Training problem is beyond activation function: Although ReLUs are used for the reported experiments, similar training characteristics for units' activations, weights and gradients are observed when linear units are used. As such, the training problems of PlainNets are beyond the type

of activation function employed.

(b) Successful training of PlainNet: We posit that working solutions for the successful training of very deep PlainNets would be related to weights regularization so that the largest eigenvalues are only slightly larger than others; this would alleviate information loss in the hidden representations that results from the domination of small eigenvalues by extremely large eigenvalues.

(c) Problem with the DiracNet [56]: The DiracNet claims to reslove the training problem of very deep PlainNet via a special initialization scheme. The weight initialization of an L-layer DiracNet, parameterized by $\widetilde{\boldsymbol{W}}^l \in \mathbb{R}^{n\times n} : 0 \leq l \leq L$, is

$$\widetilde{\boldsymbol{W}}^l = (\boldsymbol{W}^l + \text{diag}(\boldsymbol{a})) : 0 \leq l \leq L \tag{28}$$

where $\boldsymbol{W}^l$ is a randomly initialized matrix. e.g. as in [32]; and $\text{diag}(\boldsymbol{a}) \in \mathbb{R}^{n\times n}$ is the diagnoal matrix with diagonal entries $\boldsymbol{a} \in \mathbb{R}^n$. Subsequently, the output in the final layer of the DiracNet, $\widetilde{\boldsymbol{h}(\boldsymbol{x})}^L$, can be expressed as

$$\widetilde{\boldsymbol{h}(\boldsymbol{x})}^L = \prod_{l=1}^{L} \widetilde{\boldsymbol{W}}^l \boldsymbol{x}. \tag{29}$$

The DiracNet, which initializes $\boldsymbol{a} = \mathbf{1}$ mimics the ResNet at the start of training considering that (28) temporarily holds, since $\text{diag}(\boldsymbol{a})$ is the identity matrix. That is, the DiracNet behaves like it uses skip connections of identity mapping at the start of training. *Unfortunately, as training progresses, $\widetilde{\boldsymbol{W}}^l$ in* (28) *can deviate from the desired initialized setting $\boldsymbol{a} = \mathbf{1}$, since all the entries in $\widetilde{\boldsymbol{W}}^l$ are free parameters; there is no constraint on diag$(\boldsymbol{a})$ in* (28) *so that it remains the identity matrix during training*. As such, it is not surprising that the DiracNet's performance in [56] is worse than the ResNet, and the ResNet is the preferred model for many tasks based on performance. In contrast, the DiracNet is rarely used in the literature. Interestingly, keeping the hidden layer representations close to the input has been posited as crucial for the successful training of very deep models [39]. We note that the DiracNet invariably violates $\text{diag}(\boldsymbol{a}) = \mathbf{I}$ in (28) after several training iterations so that $\widetilde{\boldsymbol{h}(\boldsymbol{x})}^L$ is far from $\boldsymbol{x}$ in (29), and thus optimization becomes difficult.

(d) Skip connections are immutable constraints on the geometric structure of layer weights: The physical skip connections seen in the ResNet [17] and ResNeXt [21] impose an immutable constraint on the geometric structure of layer weights, $(\boldsymbol{W}^l + \mathbf{I})$, *where $\mathbf{I}$ is always an identity matrix given that the entries of $\mathbf{I}$ are not free parameters by model construction, and hence does not change during training*. Consequently, $\boldsymbol{h}(\boldsymbol{x})^L$ stays close to the input $\boldsymbol{x}$ in (17), and hence optimization successfully converges as posited in [39].

7. Conclusion

Remarkable results on various computer vision tasks have been obtained using DNNs with skip connections. However, their mode of operation as compared to DNNs without skip connections (PlainNets) remains largely elusive. This paper studies the unique properties of models with skip connections that eases optimization and improves generalization. Namely, important properties

such as the information fidelity (i.e. usefulness) of computed hidden representation basis and error gradient, and stability of solutions based on the conditioning of learned model parameters are investigated. Our results reveal that the training problem of very deep neural networks is beyond the saturation of hidden units that is caused by non-linear activation function. Without skip connections, very deep models exhibit singularity problems that plague optimization. However, with skip connections, the singularity problem is circumnavigated so that optimization successfully converges.

Acknowledgment

This work was funded by the National Research Fund (FNR), Luxembourg, under the project references R-AGR-0424-05-D/Bjorn Ottersten and CPPP17/IS/11643091/IDform/Aouada.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Appendix

A1. Proof of Proposition 1

Considering $\boldsymbol{W} = [\boldsymbol{w}_i]_{i=1}^{n}$, where $\boldsymbol{w}_i$ is the i-th vector in $\boldsymbol{W}$, and its elements are randomly sampled from a continuous distribution (i.e. uniform or Gaussian). Given $\boldsymbol{w}_i$, the matrix $\boldsymbol{W}$ excluding $\boldsymbol{w}_i$ is denoted $\boldsymbol{W}_{-i}$; thus, the span of $\boldsymbol{W}_i$ is $\boldsymbol{W}_{-i}^{span} = span(\boldsymbol{w}_1, .., \boldsymbol{w}_{n-1})$. Consequently, proofing Proposition 1 translates to showing that any given $\boldsymbol{w}_i$ does not lie in the span of $\boldsymbol{W}_{-i}^{span}$; that is, $\boldsymbol{w}_i$ and $\boldsymbol{W}_{-i}^{span}$ are linearly independent.
First, it is easy to note that $P(\boldsymbol{w}_1 \neq 0) = 1$, since the Lebesgue measure of a singleton set is zero [57]; thus, $P(w_1 \notin \boldsymbol{W}_{-i}^{span}) = 1$. Furthermore, for $p \in \{2, ..., n-1\}$, let $\boldsymbol{W}_{-1,p} = [\boldsymbol{w}_1, ..., \boldsymbol{w}_p]$ be the matrix whose columns are the first p vectors, excluding vector $\boldsymbol{w}_i$, such that the columns of $\boldsymbol{W}_{-1,p}$ are linearly independent with a probability of 1. Therefore, we can state with a probablity of 1 that $\boldsymbol{W}_{-1,p}$ spans the p-dimensional subspace of $\mathbb{R}^n$, and thus also has a Lebesgue measure of zero. Interestingly, $\boldsymbol{w_{p+1}}$ resides on $\mathbb{R}^n$, and hence is on the exterior of the subspace with a probability of 1. □

A2. Expanded derivation of (12)

Herein, we provide the expanded derivation of (12) in the main paper, which is based on Assumption 1.
Let the basis vectors formed by the eigenvectors of $\boldsymbol{W}^l$ (for $1 \leq l \leq L$) in $\mathbb{R}^n$ be $\boldsymbol{V} = [\boldsymbol{v}_1, ..., \boldsymbol{v}_n]$, where $\boldsymbol{V}$ is a unitary matrix. Furthermore, given matrices $\boldsymbol{W}^a$ and $\boldsymbol{W}^b$ that are diagonizable, we have

$$\boldsymbol{V}^{-1}\boldsymbol{W}^a\boldsymbol{V} = \boldsymbol{D}^a = \begin{bmatrix} \lambda_1^a & & \\ & \ddots & \\ & & \lambda_n^a \end{bmatrix}, \text{ and} \tag{30}$$

$$\boldsymbol{V}^{-1}\boldsymbol{W}^b\boldsymbol{V} = \boldsymbol{D}^b = \begin{bmatrix} \lambda_1^b & & \\ & \ddots & \\ & & \lambda_n^b \end{bmatrix}. \tag{31}$$

Since $\boldsymbol{V}$ is unitary, we can write the following

$$\boldsymbol{W}^a = \boldsymbol{V}\boldsymbol{D}^a\boldsymbol{V}^{-1}, \text{ and} \tag{32}$$

$$\boldsymbol{W}^b = \boldsymbol{V}\boldsymbol{D}^b\boldsymbol{V}^{-1}. \tag{33}$$

Multiplying (32) by (33) gives

$$\boldsymbol{W}^a\boldsymbol{W}^b = \boldsymbol{V}\boldsymbol{D}^a\boldsymbol{V}^{-1}\boldsymbol{V}\boldsymbol{D}^b\boldsymbol{V}^{-1} = \boldsymbol{V}\boldsymbol{D}^a\boldsymbol{D}^b\boldsymbol{V}^{-1}. \tag{34}$$

For $\boldsymbol{x} = \sum_{i=1}^{n} \alpha_i \boldsymbol{v_i},\ \alpha_i \in \mathbb{R}$; where $\boldsymbol{\alpha} = [\alpha_1, ..., \alpha_n]^T$, we can write

$$\boldsymbol{x} = \boldsymbol{V}\boldsymbol{\alpha}, \tag{35}$$

The output of a linear L-layer DNN with weight matrices $\{\boldsymbol{W}^L, \boldsymbol{W}^{L-1} \cdots \boldsymbol{W}^l \cdots \boldsymbol{W}^1\}$ can be written as

$$\boldsymbol{h}(\boldsymbol{x})^L = \prod_{l=1}^{L} \boldsymbol{W}^l\boldsymbol{x} = \boldsymbol{W}^L\boldsymbol{W}^{L-1} \cdots \boldsymbol{W}^l \cdots \boldsymbol{W}^1\boldsymbol{x}. \tag{36}$$

Putting (35) in (36), and applying (34) yields

$$\boldsymbol{h}(\boldsymbol{x})^L = [\boldsymbol{V}\boldsymbol{D}^L\boldsymbol{V}^{-1}\boldsymbol{V}\boldsymbol{D}^{L-1}\boldsymbol{V}^{-1}...\boldsymbol{V}^1\boldsymbol{D}^1\boldsymbol{V}^{-1}]\boldsymbol{V}\boldsymbol{\alpha}. \tag{37}$$

Tidying up (37) results in

$$\boldsymbol{h}(\boldsymbol{x})^L = \boldsymbol{V}[\prod_{l=1}^{L} \boldsymbol{D}^l]\boldsymbol{\alpha}, \tag{38}$$

which can be expressed as

$$\boldsymbol{h}(\boldsymbol{x})^L = [\boldsymbol{v}_1, ..., \boldsymbol{v}_n] \begin{bmatrix} \prod_{l=1}^{L} \lambda_1^l & & \\ & \ddots & \\ & & \prod_{l=1}^{L} \lambda_n^l \end{bmatrix} \begin{bmatrix} \alpha_1 \\ \vdots \\ \alpha_n \end{bmatrix}, \text{ and} \tag{39}$$

$$\boldsymbol{h}(\boldsymbol{x})^L = [(\prod_{l=1}^{L} \lambda_1^l)\boldsymbol{v}_1, ..., (\prod_{l=1}^{L} \lambda_n^l)\boldsymbol{v}_n] \begin{bmatrix} \alpha_1 \\ \vdots \\ \alpha_n \end{bmatrix}. \tag{40}$$

The derivation is completed by writing (40) as

$$\boldsymbol{h}(\boldsymbol{x})^L = \prod_{l=1}^{L} \lambda_1^l\alpha_1\boldsymbol{v}_1 + ... + \prod_{l=1}^{L} \lambda_n^l\alpha_n\boldsymbol{v}_n, \tag{41}$$

and

$$\boldsymbol{h}(\boldsymbol{x})^L = \sum_{i=1}^{n} \alpha_i\boldsymbol{v}_i \prod_{l=1}^{L} \lambda_i^l. \ \square \tag{42}$$

A3. Proof of Theorem 3

First, we lump the weight matrices in (5) of the main paper for $1 \leq k \leq m$ to

$$\boldsymbol{W}^{l,k} = \boldsymbol{W}_b^{l,k}\boldsymbol{W}_b^{l-1,k}, \tag{43}$$

so that we can write the output for the ResNeXt as

$$\boldsymbol{h}(\boldsymbol{x})^L = \prod_{l=1}^{L}\left(\sum_{k=1}^{m}\boldsymbol{W}^{l,k} + \mathbf{I}\right)\boldsymbol{x}. \tag{44}$$

Using Lemma 1 with $c = 1$ in the main paper, we have $(\boldsymbol{W}^{l,k} + \mathbf{I})\boldsymbol{v}_i = (\lambda_i^{l,k} + 1)\boldsymbol{v}_i$. Using (7) and Assumption 1 from the main paper in (15), we get

$$\boldsymbol{h}(\boldsymbol{x})^L = \sum_{i=1}^{n}\alpha_i\boldsymbol{v}_i\prod_{l=1}^{L}(\sum_{j=1}^{m}\lambda_i^{l,k} + 1). \tag{45}$$

Furthermore, let $p_i^{l,k} = \sum_{j=1}^{m}\lambda_i^{l,k}$, such that (45) can now be written as

$$\boldsymbol{h}(\boldsymbol{x})^L = \sum_{i=1}^{n}\alpha_i\boldsymbol{v}_i\prod_{l=1}^{L}(p_i^{l,k} + 1). \tag{46}$$

Again, using Axiom 1 in the main paper gives

$$\begin{aligned}\boldsymbol{h}(\boldsymbol{x})^L = \sum_{i=1}^{n}\alpha_i\boldsymbol{v}_i + \left(\sum_{l=1}^{L}p_1^{l,k} + \cdots + \prod_{l=1}^{L}p_1^{l,k}\right)\Bigg\{\alpha_1\boldsymbol{v}_1 + \\ \sum_{i=2}^{n}\alpha_i\boldsymbol{v}_i\left(\frac{\sum_{l=1}^{L}p_i^{l,k} + \cdots + \prod_{l=1}^{L}p_i^{l,k}}{\sum_{l=1}^{L}p_1^{l,k} + \cdots + \prod_{l=1}^{L}p_1^{l,k}}\right)\Bigg\}.\end{aligned} \tag{47}$$

In conclusion, when $L \gg 1$, and $\left|p_1^{l,k}\right| > \left|p_i^{l,k}\right| : 2 \leq i \leq n$, as expected for eigenvalues to (47), and substituting $p_i^{l,k} = \sum_{j=1}^{m}\lambda_i^{l,k}$ finalizes the proof . □

A4. Proof of Proposition 3

Let us consider a simple problem, $\boldsymbol{Y} = \boldsymbol{W}\boldsymbol{X}$, where $\boldsymbol{W} \in \mathbb{R}^{n\times n}$, $\boldsymbol{X} \in \mathbb{R}^{n\times N}$ and $\boldsymbol{Y} \in \mathbb{R}^{n\times N}$. Our goal is to estimate the solution, $\boldsymbol{W}$, given that $\boldsymbol{X}^\dagger$ is the pseudoinvrese of X. Furthermore, let a *small* perturbation of $\Delta\boldsymbol{X}$ result in a *small* solution perturbation, $\Delta\boldsymbol{W}$, so that

$$\boldsymbol{Y} = (\boldsymbol{W} + \Delta\boldsymbol{W})(\boldsymbol{X} + \Delta\boldsymbol{X}). \tag{48}$$

Given that $\Delta\boldsymbol{W}\Delta\boldsymbol{X} \approx 0$ and $\boldsymbol{Y} = \boldsymbol{W}\boldsymbol{X}$, (48) results in

$$\boldsymbol{W}^{-1}\Delta\boldsymbol{W} = -\Delta\boldsymbol{X}\boldsymbol{X}^\dagger. \tag{49}$$

Applying Cauchy-Schwarz inequality to (49) gives

$$\frac{\| \Delta \boldsymbol{W} \|}{\| \boldsymbol{W} \|} \leq \| \Delta \boldsymbol{X} \| \| \boldsymbol{X}^{\dagger} \| \leq \| \Delta \boldsymbol{X} \| \| \boldsymbol{X}^{\dagger} \| \frac{\| \boldsymbol{X} \|}{\| \boldsymbol{X} \|}. \tag{50}$$

That is

$$\frac{\| \Delta \boldsymbol{W} \|}{\| \boldsymbol{W} \|} \leq \kappa(\boldsymbol{X}) \frac{\| \Delta \boldsymbol{X} \|}{\| \boldsymbol{X} \|}, \tag{51}$$

where $\kappa(\boldsymbol{X}) \approx \| \boldsymbol{X}^{\dagger} \| \| \boldsymbol{X} \|$. □

References

[1] N. Hussein, E. Gavves, A. W. Smeulders, Timeception for complex action recognition, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2019, pp. 254–263.

[2] Y. Rao, J. Lu, J. Zhou, Spherical fractal convolutional neural networks for point cloud recognition, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2019, pp. 452–460.

[3] X. Xu, Y. Ma, W. Sun, Towards real scene super-resolution with raw images, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2019, pp. 1723–1731.

[4] H. Cheng, D. Lian, S. Gao, Y. Geng, Evaluating capability of deep neural networks for image classification via information plane, in: Proceedings of the European Conference on Computer Vision (ECCV), 2018, pp. 168–182.

[5] X. Li, S. Chen, X. Hu, J. Yang, Understanding the disharmony between dropout and batch normalization by variance shift, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2019, pp. 2682–2690.

[6] M. Rolinek, D. Zietlow, G. Martius, Variational autoencoders pursue pca directions (by accident), in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2019, pp. 12406–12415.

[7] O. K. Oyedotun, A. El Rahman Shabayek, D. Aouada, B. Ottersten, Highway network block with gates constraints for training very deep networks, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops, 2018, pp. 1658–1667.

[8] D. Han, J. Kim, J. Kim, Deep pyramidal residual networks, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2017, pp. 5927–5935.

[9] O. K. Oyedotun, A. Khashman, Deep learning in vision-based static hand gesture recognition, Neural Computing and Applications 28 (12) (2017) 3941–3951.

[10] D. CireşAn, U. Meier, J. Masci, J. Schmidhuber, Multi-column deep neural network for traffic sign classification, Neural networks 32 (2012) 333–338.

[11] M. Lin, Q. Chen, S. Yan, Network in network, arXiv preprint arXiv:1312.4400.

[12] A. Krizhevsky, I. Sutskever, G. E. Hinton, Imagenet classification with deep convolutional neural networks, in: Advances in neural information processing systems, 2012, pp. 1097–1105.

[13] K. Simonyan, A. Zisserman, Very deep convolutional networks for large-scale image recognition, in: International Conference on Learning Representations, 2015, pp. 1–14.

[14] M. Bianchini, F. Scarselli, On the complexity of neural network classifiers: A comparison between shallow and deep architectures, IEEE transactions on neural networks and learning systems 25 (8) (2014) 1553–1565.

[15] O. Delalleau, Y. Bengio, Shallow vs. deep sum-product networks, in: Advances in Neural Information Processing Systems, 2011, pp. 666–674.

[16] Z. Wu, C. Shen, A. Van Den Hengel, Wider or deeper: Revisiting the resnet model for visual recognition, Pattern Recognition 90 (2019) 119–133.

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

[18] R. K. Srivastava, K. Greff, J. Schmidhuber, Training very deep networks, in: Advances in neural information processing systems, 2015, pp. 2377–2385.

[19] K. He, X. Zhang, S. Ren, J. Sun, Identity mappings in deep residual networks, in: European conference on computer vision, Springer, 2016, pp. 630–645.
[20] G. Larsson, M. Maire, G. Shakhnarovich, Fractalnet: Ultra-deep neural networks without residuals, in: International Conference on Learning Representations, 2017, pp. 1–11.
[21] S. Xie, R. Girshick, P. Dollár, Z. Tu, K. He, Aggregated residual transformations for deep neural networks, in: Proceedings of the IEEE conference on computer vision and pattern recognition, 2017, pp. 1492–1500.
[22] X. Zhang, Z. Li, C. Change Loy, D. Lin, Polynet: A pursuit of structural diversity in very deep networks, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2017, pp. 718–726.
[23] G. Huang, Z. Liu, L. Van Der Maaten, K. Q. Weinberger, Densely connected convolutional networks, in: Proceedings of the IEEE conference on computer vision and pattern recognition, 2017, pp. 4700–4708.
[24] C. Szegedy, S. Ioffe, V. Vanhoucke, A. A. Alemi, Inception-v4, inception-resnet and the impact of residual connections on learning, in: Thirty-First AAAI Conference on Artificial Intelligence, 2017.
[25] T. E. Booth, Power iteration method for the several largest eigenvalues and eigenfunctions, Nuclear science and engineering 154 (1) (2006) 48–62.
[26] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, A. Rabinovich, Going deeper with convolutions, in: Proceedings of the IEEE conference on computer vision and pattern recognition, 2015, pp. 1–9.
[27] A. Mollahosseini, D. Chan, M. H. Mahoor, Going deeper in facial expression recognition using deep neural networks, in: 2016 IEEE Winter conference on applications of computer vision (WACV), IEEE, 2016, pp. 1–10.
[28] H. Lee, H. Kwon, Going deeper with contextual cnn for hyperspectral image classification, IEEE Transactions on Image Processing 26 (10) (2017) 4843–4855.
[29] A. Shrivastava, R. Sukthankar, J. Malik, A. Gupta, Beyond skip connections: Top-down modulation for object detection, arXiv preprint arXiv:1612.06851.
[30] T. Fernando, S. Denman, S. Sridharan, C. Fookes, Going deeper: Autonomous steering with neural memory networks, in: Proceedings of the IEEE International Conference on Computer Vision, 2017, pp. 214–221.
[31] O. K. Oyedotun, A. E. R. Shabayek, D. Aouada, B. Ottersten, Training very deep networks via residual learning with stochastic input shortcut connections, in: International Conference on Neural Information Processing, Springer, 2017, pp. 23–33.
[32] K. He, X. Zhang, S. Ren, J. Sun, Delving deep into rectifiers: Surpassing human-level performance on imagenet classification, in: Proceedings of the IEEE international conference on computer vision, 2015, pp. 1026–1034.
[33] X. Glorot, Y. Bengio, Understanding the difficulty of training deep feedforward neural networks, in: Proceedings of the thirteenth international conference on artificial intelligence and statistics, 2010, pp. 249–256.
[34] S. Ioffe, C. Szegedy, Batch normalization: Accelerating deep network training by reducing internal covariate shift, in: International Conference on Machine Learning, 2015, pp. 448–456.
[35] K. Zhang, M. Sun, T. X. Han, X. Yuan, L. Guo, T. Liu, Residual networks of residual networks: Multilevel residual networks, IEEE Transactions on Circuits and Systems for Video Technology 28 (6) (2017) 1303–1314.
[36] Y. Chen, J. Li, H. Xiao, X. Jin, S. Yan, J. Feng, Dual path networks, in: Advances in Neural Information Processing Systems, 2017, pp. 4467–4475.
[37] D. Balduzzi, M. Frean, L. Leary, J. Lewis, K. W.-D. Ma, B. McWilliams, The shattered gradients problem: If resnets are the answer, then what is the question?, in: Proceedings of the 34th International Conference on Machine Learning-Volume 70, JMLR. org, 2017, pp. 342–350.
[38] A. Veit, M. J. Wilber, S. Belongie, Residual networks behave like ensembles of relatively shallow networks, in: Advances in neural information processing systems, 2016, pp. 550–558.
[39] K. Greff, R. K. Srivastava, J. Schmidhuber, Highway and residual networks learn unrolled iterative estimation, in: International Conference on Learning Representations, 2017.
[40] J. Zhang, C. Ma, J. Liu, G. Shi, Penetrating the influence of regularizations on neural network based on information bottleneck theory, Neurocomputing 393 (2020) 76–82.
[41] Y. Zhou, Y. Liang, Critical points of linear neural networks: Analytical forms and landscape properties, in: International Conference on Learning Representations, 2019.
[42] S. Sonoda, N. Murata, Transport analysis of infinitely deep neural network, JMLR 20 (1) (2019) 31–82.
[43] A. M. Saxe, J. L. McClelland, S. Ganguli, Exact solutions to the nonlinear dynamics of learning in deep linear

neural networks, in: International Conference on Learning Representations, 2014, pp. 1–21.
[44] K. Kawaguchi, Deep learning without poor local minima, in: Advances in neural information processing systems, 2016, pp. 586–594.
[45] Q. Nguyen, M. Hein, Optimization landscape and expressivity of deep cnns, in: International Conference on Machine Learning, 2018, pp. 3727–3736.
[46] S. Arora, N. Cohen, N. Golowich, W. Hu, A convergence analysis of gradient descent for deep linear neural networks, in: International Conference on Learning Representations, 2019.
[47] S. Arora, N. Cohen, W. Hu, Y. Luo, Implicit regularization in deep matrix factorization, in: Advances in Neural Information Processing Systems, 2019, pp. 7413–7424.
[48] D. Duvenaud, O. Rippel, R. Adams, Z. Ghahramani, Avoiding pathologies in very deep networks, in: Artificial Intelligence and Statistics, 2014, pp. 202–210.
[49] L. Elden, Algorithms for the regularization of ill-conditioned least squares problems, BIT Numerical Mathematics 17 (2) (1977) 134–145.
[50] S. Chen, Local regularization assisted orthogonal least squares regression, Neurocomputing 69 (4-6) (2006) 559–585.
[51] Y. LeCun, C. Cortes, Mnist handwritten digit database, `http://yann.lecun.com/exdb/mnist/` (Last accessed, October. 2019).
[52] A. Krizhevsky, V. Nair, G. Hinton, Cifar-10, cifar-100 (canadian institute for advanced research), `http://www.cs.toronto.edu/~kriz/cifar.html` (Last accessed, October. 2019).
[53] H. S. J. K. S. S. S. M. Z. H. A. K. A. K. M. B. A. C. B. Olga Russakovsky, Jia Deng*, L. Fei-Fei, Imagenet large scale visual recognition challenge, International Journal of Computer Vision (IJCV) (December 2015) 211–252.
[54] S. Zagoruyko, N. Komodakis, Wide residual networks, in: British Machine Vision Conference, 2016, pp. 35–67.
[55] G. Bergqvist, E. G. Larsson, The higher-order singular value decomposition: Theory and an application [lecture notes], IEEE Signal Processing Magazine 27 (3) (2010) 151–154.
[56] S. Zagoruyko, N. Komodakis, Diracnets: Training very deep neural networks without skip-connections, arXiv preprint arXiv:1706.00388.
[57] J. M. Briggs, T. Schaffter, Measure and cardinality, The American Mathematical Monthly 86 (10) (1979) 852–855.