Why is Everyone Training Very Deep Neural Network with Skip Connections?

Oyebade K. Oyedotun, *Student Member, IEEE,* Kassem Al Ismaeil, *Member, IEEE,* and Djamila Aouada, *Senior Member, IEEE,*

***Abstract*—Recent deep neural networks (DNNs) with several layers of feature representations rely on some form of skip connections to simultaneously circumnavigate optimization problems and improve generalization performance. However, the operations of these models are still not clearly understood, especially in comparison to DNNs without skip connections referred to as PlainNets that are absolutely untrainable beyond some depth. As such, the exposition of this paper is the theoretical analysis of the role of skip connections in training very DNNs using concepts from linear algebra and random matrix theory. In comparison with PlainNets, the results of our investigation directly unravel the following (i) why DNNs with skip connections are easier to optimize (ii) why DNNs with skip connections exhibit improved generalization. Our investigation results concretely show that the hidden representations of PlainNets progressively suffer from information loss via singularity problems with depth increase, and thus making their optimization difficult. In contrast, as model depth increases, the hidden representations of DNNs with skip connections circumnavigate singularity problems to retain full information that reflects in improved optimization and generalization. For theoretical analysis, this paper studies in relation to PlainNets two popular skip-connection based DNNs that are residual networks (ResNets) and residual network with aggregated features (ResNeXt).**

***Index Terms*—Very deep neural network, skip connection, optimization, generalization**

I. Introduction

Deep neural networks (DNNs) have given remarkable results on various learning tasks. On one hand, the capability to automatically learn relevant features in an end-to-end fashion from different datasets [1], [2] makes DNNs quite appealing as opposed to the traditional method of handcrafting features, and then employing a simple classifier. On the other hand, DNNs are often considered as black-box models due to the limited knowledge of what has been learned in the different layers [3], [4], [5], and why simple gradient descent is generally able to find decent solutions for such highly non-convex optimization problems [6], [7]. Nevertheless, considerable progress has been made in different aspects of understanding the operation of the DNNs. Some of these include various visualization methods [8], [9], [10], [11], saliency tracking and pruning for identifying important parameters [12], [13], analytical studies of model representational capacity [14], [15], [16], and works on model optimization [17], [18] and generalization [19], [20], [21].

O.K. Oyebade, K. Ismaeil and D. Aouada are with the Interdisciplinary Centre for Security, Reliability and Trust (SnT), University of Luxembourg, L-1855 Luxembourg.

E-mail: {oyebade.oyedotun, kassem.alismaeil, djamila.aouada}@uni.lu.

Manuscript received April 10, 2020; revised XX XX, 20XX.

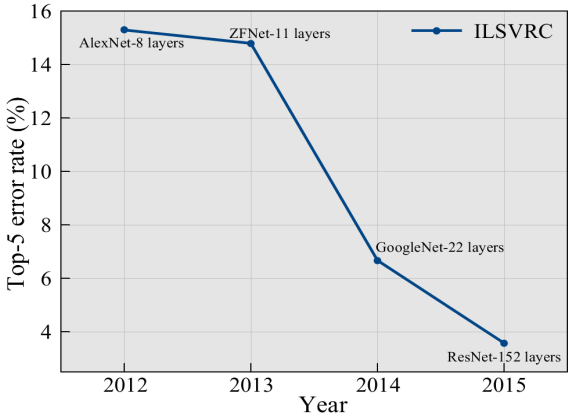

Fig. 1: The impact of depth on achieved error rate (%) for the ImageNet-2012 dataset [25]

In the early period of DNNs, most models employed two to four feature representation layers to achieve improved results [22], [23]. However, there has been consistent desire ever since to train deeper models, given that there seems to be a positive correlation between model depth and generalization performance based on empirical studies. In fact, one can observe the evolution of the state-of-the-art results on the popular ImageNet-2012 classification dataset [24]; better results have been reported by using deeper models; see Fig. 1 [25]. Interestingly, there are theoretical works [26], [27], [28] that substantiate the impact of depth for function approximation capacity of DNNs. It is important to note in Fig. 1 that both AlexNet and ZFNet are PlainNets, while both GoogleNet and ResNet use some form of skip connections. Generally, training PlainNets with few number of layers (i.e. typically one to ten layers) is not problematic. However, when model depth is extended beyond 10 layers, training difficulty can ensue. This problem typically worsens with depth increase, and sometimes even the training set cannot be fitted [29], [30]. For example, the work in [31] reported that though the VGG-11 model (having 11 layers) could be successfully trained from scratch, there was optimization failure for the VGG-13 model (having 13 layers), VGG-16 model (having 16 layers) and VGG-19 model (having 19 layers) when training from scratch. As such, the VGG-13 model was successfully trained by initializing its first 11 layers with the weights of the already trained VGG-11 model. Furthermore, the VGG-16 model was trained by initializing its layers with the weights of the already trained VGG-13. Similarly, the successful training of the VGG-19 model relied

on initializing its layers with the weights of the already trained VGG-16 model. Therefore, recent architectures with over 15 layers employ some form of skip connections, where the outputs of preceding layers are connected (e.g. via summation or concatenation) to later layers for tackling training problems. In fact, the success of this approach has resulted in a proliferation of DNN models with different forms of skip connections, as seen in Residual Network (ResNet) [29], [32], FractalNet [33], ResNeXt [34], PolyNet [35], DenseNet [36] and Inception-ResNet [37]. Although there is abundant empirical evidence that skip connections alleviate training problems and improve model generalization, a concrete explanation for this success is still lacking in the deep learning community. Moreover, it is arguable that simply observing that 'model A' performs better than 'model B' fosters an incomplete knowledge; it is crucial to understand the unique characteristics of 'model A' that result in performance improvement as compared to 'model B'.

In this paper, our main exposition is the theoretical and experimental analysis of DNNs that employ skip connections for successful training in relation to PlainNets. Namely, we borrow several concepts from linear algebra and random matrix theory to posit new interpretations for the role of skip connections in circumnavigating optimization problems and improving model generalization. In our theoretical treatment, we consider a class of DNNs that employ skip connections of identity mappings, and the summation of preceding layers' outputs with the current layer. Specifically, we consider two popular and extremely successful models in the literature, which are the ResNet [29] and residual network with aggregated features (ResNeXt) [34]. Our main contributions in this paper are the following:

1) Analyse theoretically the optimization characteristics of very deep DNNs based on the singularity of the hidden representations. Our approach leverages several aspects of linear algebra and random matrix theory.
2) Establish the connection between the singularity of hidden representations and the singularity of computed error gradients and weight updates that contribute to the optimization condition of DNNs during backpropagation.
3) Show that the condition of hidden representations of very deep DNNs reflects on their generalization capabilities. In addition, we show concretely for the first time in the literature why the ResNeXt [34] mostly generalizes better than the ResNet [29].
4) Provide extensive experiments to corroborate theoretical results by using benchmarking datasets such as MNIST, CIFAR-10, CIFAR-100 and ImageNet.

The remainder of this paper is organized as follows. Section II discusses related works. In Section III, the relevance of studying linear DNNs is discussed as background, and a formal introduction to PlainNet, ResNet and ResNeXt models is provided as preliminaries. The proposed theoretical study of the role of skip connections for optimizing the different DNNs is presented in Section IV. Section V relates how skip connections impact model generalization. Extensive experimental results along with discussions are given in Section VI. Finally, the paper is concluded in Section VII.

II. Related work

Training PlainNets with several layers usually results in optimization problems [29], [30]. Generally, training problems can be observed when model depth exceeds 10 layers depending on the specific task [38]. Although various weights initialization schemes [39], [40] and batch normalization [41] alleviate the training problems, they do not resolve it, as optimizing PlainNets becomes absolutely impossible beyond a certain depth [38]; that is, very poor performance is obtained even on the training set. Interestingly, it has become well-known that the training problems of very deep PlainNets can be resolved by employing skip connections that connect the outputs of preceding layers to the later layers [29], [30]. Some of the popular DNNs, which rely on skip connections for successful training include, the ResNet [29], ResNeXt [34], highway network (HwNet) [30], densely connected network (DenseNet) [36], resnet of resnet (RoR) [42], dual path network (DPN) [43], PolyNet [35] and Inception-ResNet [37]. It is surprising that despite the success of the aforementioned models on different challenging tasks, a concrete account of their operation in relation to how they circumnavigate optimization problems, and on top of that achieve improved generalization as compared to PlainNets is still lacking in the deep learning community.

The learning attributes of the ResNet were studied in [44], where the problem of shattered gradients was investigated based on carefully designed experiments. Using auto correlation function results, the gradients of the 1-layer (i.e. shallow) PlainNet was found to resemble brown noise, and thus allow successful optimization. In contrast, the gradients of the 24-layer (i.e. very deep) PlainNet was found to resemble white noise, and hence very problematic for optimization. Fascinatingly, it was observed that the gradients of the 50-layer (i.e. very deep) ResNet falls between the brown noise and white noise. As such, [44] concluded that the gradients of ResNet are resistant to shattering; that is, they are well-structured, and therefore the ResNet is trainable even with several layers. Furthermore, it was found that the training of DNNs become more difficult as the gradients' structure transit from brown noise to white noise; this occurred as the DNNs became deeper. The work in [45] used an unrolled view to argue that the ResNet operates like an ensemble of shallow neural networks, suggesting that their true depths are much lesser than their topological depths. Furthermore, [45] argued that paths via the ResNet are of varying lengths and have limited dependence on one another. This revelation indeed suggests that the ResNet seems to depart from the strictly hierarchical operation of classical DNNs, i.e., PlainNets. In addition, [45] showed that the effective depth through which error gradients flow in the ResNet is far smaller than the architectural depth; for instance, only 17 layers for a 110-layer ResNet. The same work [45] found that removing any single block of layers in a trained ResNet does not result in a catastrophic testing performance; the performance of the ResNet was observed to remain mostly the same. In contrast, it was seen that removing any single block in the VGG model (i.e. PlainNet) resulted in a catastrophic testing performance that

is expected in a DNN, where the hidden representations are strictly hierarchical and sequential. Finally, it was posited that the ResNet circumnavigates training problems by employing mostly short paths to propagate error gradients.

In a different line of investigative work, [46] proposed the unrolled iterative estimation view of features at different DNN layer stages. The work posited that the different layers in the ResNet do not compute entirely new representations as in the PlainNet. Furthermore, it was argued that the ResNet representations can be divided into stages, and groups of residual blocks mostly perform iterative refinement of similar of features in a specific stage. Subsequently, new hidden representations are computed at other stages in the ResNet. In a following work [47], it was observed that the ResNet mostly concentrate representation learning in the first few layers, and employ the later layers for iterative refinement of already extracted representations.

The discretized dynamical systems view for the ResNet was proposed in [48], [49]. Basically, each time step was viewed as a transformation, akin to the layer transformation in the classical neural network. Additional works [50], [51] in this direction explored the concept of energy conservation via Hamilton systems and other constraints for analysing the ResNet.

Similar to the aforementioned works, this paper is dedicated to studying the unique properties of the DNNs that use skip connections. Specifically, we take the ResNet [29] and ResNeXt [34] as our case studies. However, the perspective of investigation, analysis and interpretation of results are different to the discussed related works [44], [45], [46]. Furthermore, we provide explanations and new insights into the improved generalization capacity of the ResNet and ResNeXt. This is a key contribution that is missing in the literature.

III. Preliminaries

A. Relevance of linear models

It is well known that strict theoretical analysis of practical DNNs is problematic, and various simplifications are often necessary for analytical tractability. Some of these problems stem from the compositional structure of DNNs, and especially the non-linear activation functions that they employ such that theoretical analysis entails treating highly non-convex optimization problems. Consequently, many works assume the linear activation function (i.e. linear units) amongst other simplifications [52], [53], [54], [55], [56]. In fact, both linear activation function and convex loss are assumed in [57], and stricter assumptions based on the number of hidden units, data points and layers are found in [58].

Subsequently, for the ease of theoretical analysis, the linear activation function is also assumed in this paper. Interestingly, this assumption has been found to be of negligible impact, since obtained results are quite relatable to models that assume the non-linear activation functions. For instance, the relevance of the theoretical results obtained from linear DNNs is discussed in [55]. Furthermore, [54] notes that the empirical observations made in DNNs with linear activation functions generally agree with those obtained using non-linear activation functions. These observations are not surprising given that the loss function of the DNN (having 2 or more layers) with a linear activation is non-convex, similar to the DNN (having 2 or more layers) with non-linear activation function. Notwithstanding, we show that the theoretical results in this paper, which are based on linear activation function clearly agree with practical DNNs by employing the non-linear activation function (i.e. Rectified linear function) for all experiments. Interestingly, assuming a linear activation function for our theoretical analysis has the advantage of decoupling the training problems of very DNNs from the popular problem of activation function saturation [39].

B. DNNs without and with skip Connections

In this section, the transformation learned by the DNN without skip connections, PlainNet, is presented. Furthermore, the different transformations learned by the DNNs with skip connections in this paper, ResNet and ResNeXt, are given. The formalization of the distinct transformations learned by the different DNNs would be useful for the main analyses in the following sections.

1) Plain network (PlainNet): This class of DNNs are the strictly hierarchical models, where only a single path connects a current layer to the succeeding one, and there is no skip connection of any sort; see Fig. 2. Considering the input, $\boldsymbol{h}(\boldsymbol{x})^{l-2} \in \mathbb{R}^n$ (where $\boldsymbol{x} \in \mathbb{R}^n$ is the input to the DNN), for the two consecutive layer weights in Fig. 2, W_{pb}^{l} and W_{pb}^{l-1}, the block's output can be expressed as

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

where $W_{pb}^{l}, W_{pb}^{l-1} \in \mathbb{R}^{n \times n}$, and pb indicates the weights in a PlainNet block. For simplicity, the result of transformation $W_{pb}^{l} W_{pb}^{l-1}$ in the PlainNet block in (1) is lumped as $W_{p}^{l} = W_{pb}^{l} W_{pb}^{l-1}$, where $W_{p}^{l} \in \mathbb{R}^{n \times n}$, so that we now obtain

$$\boldsymbol{h}(\boldsymbol{x})^{l} = W_{p}^{l} \boldsymbol{h}(\boldsymbol{x})^{l-2}. \tag{2}$$

2) Residual network (ResNet): The Residual network (ResNet) block [29] is shown in Fig. 2, and mainly relies on skip connections of identity mappings, which connect every residual block (having 2 or 3 weight layers) to the preceding one. Given the ResNet block, we can write

$$\boldsymbol{h}(\boldsymbol{x})^{l} = W_{rb}^{l} W_{rb}^{l-1} \boldsymbol{h}(\boldsymbol{x})^{l-2} + \boldsymbol{h}(\boldsymbol{x})^{l-2}, \tag{3}$$

where $W_{rb}^{l}, W_{rb}^{l-1} \in \mathbb{R}^{n \times n}$, and rb indicates the weights in the ResNet block. In addition, considering the identity matrix $\mathbf{I} \in \mathbb{R}^{n \times n}$, factorizing (3) yields

$$\boldsymbol{h}(\boldsymbol{x})^{l} = \left(W_{rb}^{l} W_{rb}^{l-1} + \mathbf{I}\right) \boldsymbol{h}(\boldsymbol{x})^{l-2}. \tag{4}$$

Again, the result of the transformation $W_{rb}^{l} W_{rb}^{l-1}$ in the residual block in (4) is lumped as $W_{r}^{l} = W_{rb}^{l} W_{rb}^{l-1}$, where $W_{r}^{l} \in \mathbb{R}^{n \times n}$, such that (4) becomes

$$\boldsymbol{h}(\boldsymbol{x})^{l} = \left(W_{r}^{l} + \mathbf{I}\right) \boldsymbol{h}(\boldsymbol{x})^{l-2}. \tag{5}$$

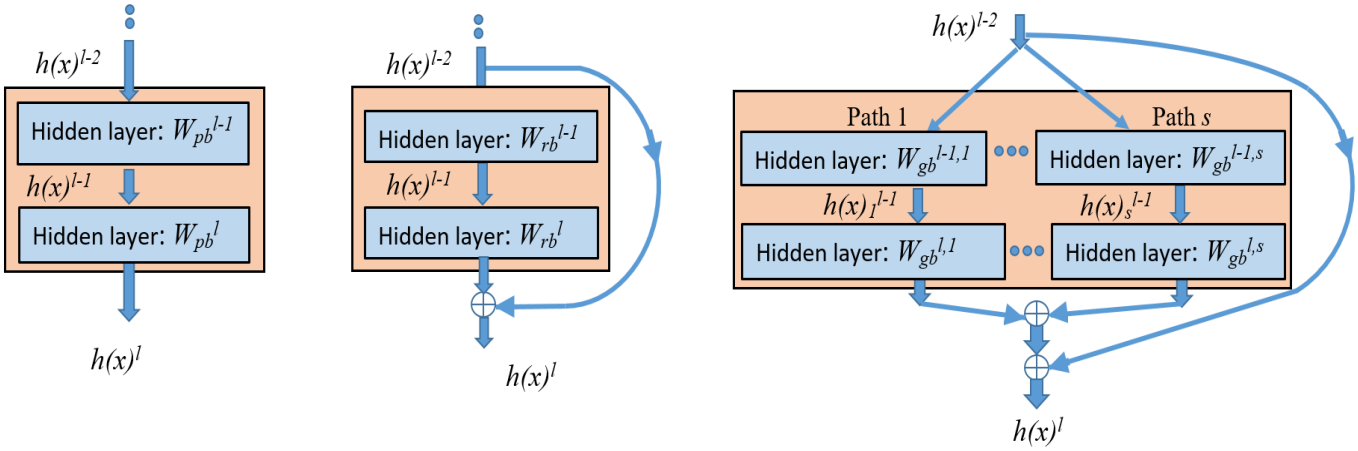

Fig. 2: Models with skip connections considered in this paper. $\oplus$ denotes addition operation. Left: PlainNet block. Middle: ResNet block. Right: ResNeXt block. Full DNNs are constructed by stacking several blocks

3) Residual network with features aggregation (ResNeXt): Residual network with features aggregation (ResNeXt) block [34] aggregates as output the summation of features learned via the different s paths of the ResNeXt block as in Fig. 2, where $W_{gb}^{l,k}, W_{gb}^{l-1,k} \in \mathbb{R}^{q\times q}$ with $k = 1, \cdots, s$, and gb indicates the weights in the ResNeXt block. Importantly, we observe from the construction of the PlainNet, ResNet [29] and ResNeXt [34] blocks in Fig. 2 that $q < n$. That is, the PlainNet and ResNet blocks use one large matrix for every transformation, as in $W_p^l \in \mathbb{R}^{n\times n}$ and $W_r^l \in \mathbb{R}^{n\times n}$, respectively; see (2) and (5). In contrast, the ResNeXt block uses many smaller s number of matrices for every transformation, as in $W_g^{l,k} \in \mathbb{R}^{q\times q}$ given in Fig. 2. Note that the condition $q < n$ will be important for analysing the subtle difference between the ResNet and ResNeXt in relation to generalization in Section V. For an input $\boldsymbol{h}(\boldsymbol{x})^{l-2} \in \mathbb{R}^q$, the output of the ResNeXt block, $\boldsymbol{h}(\boldsymbol{x})^l \in \mathbb{R}^q$, is

$$\begin{aligned}\boldsymbol{h}(\boldsymbol{x})^l =& W_{gb}^{l,s}W_{gb}^{l-1,s}\boldsymbol{h}(\boldsymbol{x})^{l-2} + \cdots + W_{gb}^{l,k}W_{gb}^{l-1,k}\boldsymbol{h}(\boldsymbol{x})^{l-2} \\ &+ \cdots + W_{gb}^{l,1}W_{gb}^{l-1,1}\boldsymbol{h}(\boldsymbol{x})^{l-2} + \boldsymbol{h}(\boldsymbol{x})^{l-2}.\end{aligned} \tag{6}$$

Similarly, for compactness, we let $W_g^{l,k} = W_{gb}^{l,k}W_{gb}^{l-1,k}$ in (6), where $W_g^{l,k} \in \mathbb{R}^{q\times q}$. Finally, using $\mathbf{I} \in \mathbb{R}^{q\times q}$ and factorizing (6) gives

$$\boldsymbol{h}(\boldsymbol{x})^l = (W_g^{l,s} + \cdots + W_g^{l,k} + \cdots + W_g^{l,1} + \mathbf{I})\boldsymbol{h}(\boldsymbol{x})^{l-2}, \tag{7}$$

which in a compact form is

$$\boldsymbol{h}(\boldsymbol{x})^l = (\sum_{k=1}^{s} W_g^{l,k} + \mathbf{I})\boldsymbol{h}(\boldsymbol{x})^{l-2}. \tag{8}$$

IV. Theoretical Analysis of Skip Connections for Optimization

Herein, we study the role of skip connections (discussed in Section III) for mitigating information loss and the resulting singularity problems that ensue in the training of very deep DNNs. However, optimization properties of the PlainNet are first given so that comparison with ResNet and ResNeXt is straightforward. The theoretical study relies on the following definition, lemma, proposition and corollary.

Definition 1: If the vector set, $\{\boldsymbol{w}_i \in \mathbb{R}^n\}_{i=1}^n$, are the columns of a non-singular matrix $W \in \mathbb{R}^{n\times n}$, then the vectors $\{\boldsymbol{w}_i\}_{i=1}^n$ are linearly independent and span $\mathbb{R}^n$.

For remaining parts of this paper, we clarify the notations relating to DNN weight matrices, W^m and $(W)^m$. Note that W^m is used to refer to the weight matrix at layer m, while $(W)^m$ is used to refer to a weight matrix W raised to power m. That is, $(W)^m$ means the weight matrix W multiplied m times.

Lemma 1: (Alexeev-Götze-Tikhomirov [59]) Given the entries of the matrix $W^l \in \mathbb{R}^{n\times n} : 1 \leq l \leq m$ are randomly drawn from a Gaussian distribution and $W \in \{W^1, \cdots, W^l, \cdots, W^m\}$ so that we have the products $Y = \prod_{l=1}^m W^l$ and $Z = (W)^m$, the asymptotic behaviour of the singular spectrum of Y is the same as that of the singular spectrum of Z.

Proof. See [59] for the proof that the limit distributions of the singular spectrum of both Y and Z is the Fuss-Catalan distribution.

Proposition 1: Given a matrix $W \in \mathbb{R}^{n\times n}$ whose column vectors, $\boldsymbol{w}_i$, are drawn from a uniform distribution or a Gaussian distribution, the probability that W is non-singular, $P(\boldsymbol{w}_i \notin W_{-i}^{span})$, is

$$P(\boldsymbol{w}_i \notin W_{-i}^{span}) = 1 \ : 1 \leq i \leq n \tag{9}$$

Proof. See Section A1 in the appendix.

Corollary 1: For an m-layer DNN, the initialization [39], [40] of all the layer weight matrices, $\{W^l \in \mathbb{R}^{n\times n}\}_{l=1}^m$, follows Proposition 1, and thus are all non-singular; using Definition 1 shows that every W^l is of rank n. Consequently, any $W \in \{W^l\}_{l=1}^m$ admits a singular value decomposition (SVD) of the form

$$W = \sum_{i=1}^{n} \sigma_i \boldsymbol{u_i}\boldsymbol{v_i}^T, \quad \sigma_i \in \mathbb{R}, \ \sigma_i > 0, \tag{10}$$

where σ_i are the singular values of W; $\boldsymbol{u_i} \in \mathbb{R}^n$ and $\boldsymbol{v_i} \in \mathbb{R}^n$ are the left and right singular vectors of W, respectively; and T denotes vector transpose.

It is known that singularity, where at least one singular value of W is exactly zero is not necessary for optimization problems to ensue. The *near-singularity* scenario, where the smallest singular value is extremely small is sufficient. This reflects in the condition number κ of W given as

$$\kappa(W) = \sigma_{max}(W)/\sigma_{min}(W), \tag{11}$$

where $\sigma_{max}(W)$ and $\sigma_{min}(W)$ are the maximum and minimum singular values of W, respectively. Problems that have $\kappa(W) \gg 1$ are commonly referred to as ill-conditioned, and the solutions obtained are typically quite unstable. The Eckart-Young theorem [60], [61] specifically addresses this as follows.

Theorem 1: (Eckart-Young [60], [61]) For a normalized non-singular matrix M (i.e. $\parallel M \parallel = 1$), its distance, d, to the set of ill-posed problems, S, is

$$d(M, S) = \frac{1}{\kappa(M)} = \frac{\sigma_{min}(M)}{\sigma_{max}(M)}. \tag{12}$$

Lemma 2: For a matrix $W \in \mathbb{R}^{n\times n}$, using SVD as in (10), $(W)^m$ (i.e. W raised to the power of m) can be written as

$$(W)^m = \sum_{i=1}^{n} \sigma_i^{2m} \boldsymbol{u_i} \boldsymbol{u_i}^T. \tag{13}$$

Proof. See Section A2 in the appendix.

Furthermore, given $\{\boldsymbol{u_i}\}_{i=1}^n$ is a set of orthonormal vectors as in (10) so that they form a basis in $\mathbb{R}^n$, we can express any $\boldsymbol{x} \in \mathbb{R}^n$ as

$$\boldsymbol{x} = \sum_{i=1}^{n} \alpha_i \boldsymbol{u_i}, \quad \alpha_i \in \mathbb{R}, \tag{14}$$

Definition 2: A collection of N individual hidden layer representations at layer l, $\boldsymbol{h(x)}_i^l \in \mathbb{R}^n$, compose the hidden representations data batch, $\boldsymbol{H(x)}^l \in \mathbb{R}^{n\times N}$. Specifically, $\boldsymbol{H(x)}^l = [\boldsymbol{h(x)}_1^l, \cdots, \boldsymbol{h(x)}_i^l, \cdots, \boldsymbol{h(x)}_N^l]$.

Our analysis starts first on the forward-pass and then on the backpropagation in relation to optimization conditions.

A. Forward-pass: hidden representations basis loss

This section studies the condition of the basis of the hidden representations learned by the different DNN models in the forward-pass phase. Particularly, we are interested in observing the loss or preservation of the input data basis as it is forward-propagated in the hidden layers.

1) Plain network (PlainNet): Given the PlainNet in Section III.B.1, we state the following theorem.

Theorem 2: For an input $\boldsymbol{x} \in \mathbb{R}^n$ to a linear m-layer PlainNet parameterized by $\theta_p = \{W_p^m \in \mathbb{R}^{n\times n}, \cdots, W_p^l \in \mathbb{R}^{n\times n}, \cdots, W_p^2 \in \mathbb{R}^{n\times n}, W_p^1 \in \mathbb{R}^{n\times n}\}$, the hidden layer output, $\boldsymbol{h(x)}^m : m \longrightarrow \infty$, is

$$\boldsymbol{h(x)}^m = \alpha_1 \sigma_{p_1}^{2m} \boldsymbol{u_1}, \tag{15}$$

where σ_{p_1} and $\boldsymbol{u}_1$ are the first singular value component and first singular vector of W_p^l, respectively; the scalar α_1 is the first component of $\{\boldsymbol{\alpha}_i\}_{i=1}^n$ as in (14).

Proof. The output of the PlainNet in the last layer m can be written as

$$\boldsymbol{h(x)}^m = \prod_{l=1}^{m} W_p^l \boldsymbol{x}. \tag{16}$$

Since we are ultimately interested in studying the singular values of $\boldsymbol{h(x)}^m : m \longrightarrow \infty$, Lemma 1 simplifies (16) to

$$\boldsymbol{h(x)}^m = (W_p)^m \boldsymbol{x}. \tag{17}$$

From Lemma 2, putting $(W_p)^m$ in (17) yields

$$\boldsymbol{h(x)}^m = \sum_{i=1}^{n} \sigma_{p_i}^{2m} \boldsymbol{u_i} \boldsymbol{u_i}^T. \, \boldsymbol{x}, \tag{18}$$

where writing out (18) gives

$$\boldsymbol{h(x)}^m = \left(\sigma_{p_1}^{2m} \boldsymbol{u_1} \boldsymbol{u_1}^T + \sum_{i=2}^{n} \sigma_{p_i}^{2m} \boldsymbol{u_i} \boldsymbol{u_i}^T \right) \boldsymbol{x}. \tag{19}$$

Subsequently, (19) is factorized so that we obtain

$$\boldsymbol{h(x)}^m = \sigma_{p_1}^{2m} \boldsymbol{u_1} \boldsymbol{u_1}^T \left(\mathbf{I} + \sum_{i=2}^{n} \frac{\sigma_{p_i}^{2m}}{\sigma_{p_1}^{2m}} (\boldsymbol{u_1} \boldsymbol{u_1}^T)^{-1} \boldsymbol{u_i} \boldsymbol{u_i}^T \right) \boldsymbol{x}. \tag{20}$$

Given that $m \longrightarrow \infty$ and $\sigma_{p_1} > \sigma_{p_i} : \; 2 \leq i \leq n$ as expected for singular values, (20) becomes

$$\boldsymbol{h(x)}^m = \sigma_{p_1}^{2m} \boldsymbol{u_1} \boldsymbol{u_1}^T \, \boldsymbol{x}. \tag{21}$$

Putting (14) in (21) gives

$$\boldsymbol{h(x)}^m = \sigma_{p_1}^{2m} \boldsymbol{u_1} \boldsymbol{u_1}^T \sum_{i=1}^{n} \alpha_i \boldsymbol{u_i}. \tag{22}$$

Finally, applying $\boldsymbol{u_i}^T \boldsymbol{u_j} = 0$ for $i \neq j$ and $\boldsymbol{u_i}^T \boldsymbol{u_i} = 1$ to (22) concludes the proof of Theorem 2. □

Remark 1: Considering the input, $\boldsymbol{x}$, only the first basis vector, $\boldsymbol{u_1}$, contributes to the computation of $\boldsymbol{h(x)}^m$ for $m \longrightarrow \infty$ due to repeated multiplication by $\{W_p^l\}_{i=1}^m$. Hence, $\boldsymbol{h(x)}^m \in \mathbb{R}^n$ for the PlainNet incurs considerable information loss. From Definition 2 and Theorem 2, the columns of a data batch for the PlainNet, $\boldsymbol{H(x)}_p^m \in \mathbb{R}^{n\times N}$, are colinear and thus $\boldsymbol{H(x)}_p^m$ exhibits singularity.

We note that using deep Gaussian process along with some assumptions, [62] arrived at a similar result as in Theorem 2. Namely, [62] concluded that the representational capacity of DNNs collapses to a single degree of freedom as $m \longrightarrow \infty$. In addition, the catastrophic peformance of the PlainNet from layer deletion as noted in [45] can be related to Theorem 2. For the worst case, where the deletion of all the m layer weights sets $\sigma_{p_1} = 0$ in Theorem 2, we obtain $\boldsymbol{h(x)}^m = 0$ so that there is an absolute collapse of the PlainNet's performance.

2) Residual network (ResNet): The next theorem relates the features conditioning of the ResNet as in Section III.B.2.

Theorem 3: For an input vector $\boldsymbol{x} \in \mathbb{R}^n$ to a linear m-layer ResNet parameterized by $\theta_r = \{W_r^m \in \mathbb{R}^{n\times n}, \cdots, W_r^l \in \mathbb{R}^{n\times n}, \cdots, W_r^2 \in \mathbb{R}^{n\times n}, W_r^1 \in \mathbb{R}^{n\times n}\}$, the hidden layer output, $\boldsymbol{h(x)}^m : m \longrightarrow \infty$, is

$$\boldsymbol{h(x)}^m = \alpha_1 \boldsymbol{u_1} [(\sigma_{r_1}^2 + 1)^m - 1] + \sum_{i=1}^{n} \alpha_i \boldsymbol{u_i}, \tag{23}$$

where σ_{r_1} and $\boldsymbol{u}_1$ are the first singular value component and first singular vector of W_r^l, respectively.

Proof. Using (5), the final output of the ResNet can be written as

$$\boldsymbol{h(x)^m} = \prod_{l=1}^{m} (W_r^l + \mathbf{I}) \boldsymbol{x}. \tag{24}$$

Again, applying Lemma 1 to (24) simplifies it to

$$\boldsymbol{h}(\boldsymbol{x})^m = (W_r + \mathbf{I})^m \boldsymbol{x}. \tag{25}$$

Expanding $(W_r + \mathbf{I})^m$ in (25) using binomial theorem, we have

$$\boldsymbol{h}(\boldsymbol{x})^m = ((W_r)^m + m(W_r)^{(m-1)} + m(m-1)(W_r)^{(m-2)}/2 + \cdots + \mathbf{I})\boldsymbol{x}. \tag{26}$$

Furthermore, using Lemma 2 for (26) gives

$$\boldsymbol{h}(\boldsymbol{x})^m = \Bigg(\sum_{i=1}^{n} \sigma_{r_i}^{2m} \boldsymbol{u_i} \boldsymbol{u_i}^T + m \sum_{i=1}^{n} \sigma_{r_i}^{2(m-1)} \boldsymbol{u_i} \boldsymbol{u_i}^T + m(m-1) \sum_{i=1}^{n} \sigma_{r_i}^{2(m-2)} \boldsymbol{u_i} \boldsymbol{u_i}^T/2 + \cdots + \mathbf{I} \Bigg) \boldsymbol{x}. \tag{27}$$

Given $m \longrightarrow \infty$ and $\sigma_{r_1} > \sigma_{r_i} : 2 \leq i \leq n$, and using (14), (27) becomes

$$\boldsymbol{h}(\boldsymbol{x})^m = \Bigg(\sigma_{r_1}^{2m} \boldsymbol{u_1} \boldsymbol{u_1}^T + m\sigma_{r_1}^{2(m-1)} \boldsymbol{u_1} \boldsymbol{u_1}^T + m(m-1)\sigma_{r_1}^{2(m-2)} \boldsymbol{u_1} \boldsymbol{u_1}^T/2 + \cdots + \mathbf{I} \Bigg) \sum_{i=1}^{n} \alpha_i \boldsymbol{u_i}. \tag{28}$$

Applying $\boldsymbol{u_i}^T \boldsymbol{u_j} = 0$ for $i \neq j$ and $\boldsymbol{u_i}^T \boldsymbol{u_i} = 1$ to (28) yields

$$\boldsymbol{h}(\boldsymbol{x})^m = \alpha_1 \sigma_{r_1}^{2m} \boldsymbol{u_1} + m\alpha_1 \sigma_{r_1}^{2(m-1)} \boldsymbol{u_1} + m(m-1)\alpha_1 \sigma_{r_1}^{2(m-2)} \boldsymbol{u_1}/2 + \cdots + \sum_{i=1}^{n} \alpha_i \boldsymbol{u_i}. \tag{29}$$

Factorizing out $\alpha_1 \boldsymbol{u_1}$ gives

$$\boldsymbol{h}(\boldsymbol{x})^m = \alpha_1 \boldsymbol{u_1} \Bigg(\sigma_{r_1}^{2m} + m\sigma_{r_1}^{2(m-1)} + m(m-1)\sigma_{r_1}^{2(m-2)}/2 + \cdots + 1 \Bigg) - \alpha_1 \boldsymbol{u_1} + \sum_{i=1}^{n} \alpha_i \boldsymbol{u_i}. \tag{30}$$

Finally, applying binomial theorem again so that (30) can be compactly written completes the proof of Theorem 3. □

Remark 2: Considering the input, $\boldsymbol{x}$, all the basis, $\{\boldsymbol{u_i}\}_{i=1}^n$ contribute to the computation of $\boldsymbol{h}(\boldsymbol{x})^m \in \mathbb{R}^n$ for $m \longrightarrow \infty$; repeated multiplication by $\{W_r^l\}_{i=1}^m$ does not cause any basis to vanish. Subsequently, $\boldsymbol{h}(\boldsymbol{x})^m$ for the ResNet retains full information. From Definition 2 and Theorem 3, the columns of a data batch from the ResNet $\boldsymbol{H}(\boldsymbol{x})_r^m \in \mathbb{R}^{n \times N}$ are distinct, and thus $\boldsymbol{H}(\boldsymbol{x})_r^m$ is non-singular.

Furthermore, the invulnerability of ResNet performance to layer deletion as noted in [45] can be explained by Theorem 3. In the worst case, where the deletion of all the m layer weights sets $\sigma_{r_1} = 0$ in Theorem 3, we obtain $\boldsymbol{h}(\boldsymbol{x})^m = \sum_{i=1}^n \alpha_i \boldsymbol{u_i}$. Hence, $\boldsymbol{h}(\boldsymbol{x})^m = \boldsymbol{x}$, and there is no catastrophic performance.

3) ResNeXt: The ResNeXt analysis is based on the discussion in Section III.B.3, and will employ the following lemmas.

Lemma 3: Let the set of s independent Gaussian random matrices be $\{X_1 \sim \mathcal{N}(\mu_1, \beta_1^2), X_2 \sim \mathcal{N}(\mu_2, \beta_2^2), \cdots, X_k \sim \mathcal{N}(\mu_k, \beta_k^2), \cdots, X_s \sim \mathcal{N}(\mu_s, \beta_s^2)\}$, and $X_s = \sum_{k=1}^s X_k$; where μ_k and β_k^2 are the mean and variance of X_k, respectively. Then, it can be shown that

$$X_s \sim \mathcal{N}(\sum_{k=1}^{s} \mu_k, \sum_{k=1}^{s} \beta_k^2), \tag{31}$$

Proof. See Section A3 in the appendix.

Namely, Lemma 3 allows us to replace $\sum_{k=1}^s W_g^{l,k}$ in (8) with a single matrix W_c^l. As such, we can expresses (8) compactly as

$$\boldsymbol{h}(\boldsymbol{x})^l = (W_c^l + \mathbf{I})\boldsymbol{h}(\boldsymbol{x})^{l-2}, \tag{32}$$

where $W_c^l \sim \mathcal{N}(\sum_{k=1}^s \mu_k^l, \sum_{k=1}^s \beta_k^2)$, and μ_k^l and β_k^2 are the mean and variance of the distribution of $W_g^{l,k}$, respectively . Note that $W_c^l \in \mathbb{R}^{q \times q}$ similar to $W_g^{l,k}$. Equipped with (32) for the ResNeXt, the next theorem is stated as follows.

Theorem 4: For the input $\boldsymbol{x} \in \mathbb{R}^n$ to a linear m-layer ResNeXt parameterized by $\theta_c = \{W_c^m \in \mathbb{R}^{q \times q}, \cdots, W_c^l \in \mathbb{R}^{q \times q}, \cdots, W_c^2 \in \mathbb{R}^{q \times q}, W_c^1 \in \mathbb{R}^{q \times n}\}$, the hidden layer output, $\boldsymbol{h}(\boldsymbol{x})^m : m \longrightarrow \infty$, is

$$\boldsymbol{h}(\boldsymbol{x})^m = \alpha_1 \boldsymbol{u_1}[(\sigma_{c_1}^2 + 1)^m - 1] + \sum_{i=1}^{q} \alpha_i \boldsymbol{u_i}, \tag{33}$$

where σ_{c_1} and $\boldsymbol{u}_1$ are the first singular value component and first singular vector of W_c^l, respectively.

Proof. From (32), following a similar proof for the ResNet given in Theorem 3, we arrive at an analogous expression for the ResNeXt. □

Remark 3: Similar to the ResNet, it is seen that $\{\boldsymbol{u_i}\}_{i=1}^n$ contributes to the computation of $\boldsymbol{h}(\boldsymbol{x})^m \in \mathbb{R}^q$ for $m \longrightarrow \infty$, as repeated multiplication by $\{W_c^l\}_{i=1}^m$ does not cause any basis to vanish. Hence, the ResNeXt retains full information. Again, using Definition 2 and Theorem 4, the columns of a data batch for the ResNeXt $\boldsymbol{H}(\boldsymbol{x})_c^m \in \mathbb{R}^{q \times N}$ are different, and therefore there is no singularity.

Although the vulnerability of ResNeXt's performance to layer deletion has not been empirically studied in any work to the best of our knowledge, Theorem 4 that is analogous to Theorem 3, theoretically shows that the ResNeXt behaves in a similar way to the ResNet. Therefore, the ResNeXt circumnavigates catastrophic performance resulting from deleting m layer weights. That is, $\sigma_{c_1} = 0$ in Theorem 4, so that $\boldsymbol{h}(\boldsymbol{x})^m = \sum_{i=1}^q \alpha_i \boldsymbol{u_i}$, and thus $\boldsymbol{h}(\boldsymbol{x})^m = \boldsymbol{x}$.

B. Backpropagation: singularity of error gradients

DNN training generally relies on the gradient descent algorithm, where local error gradients are responsible for driving optimization. As such, the condition of computed error gradients impact the successful convergence of DNN optimization. Specifically, if the error gradients are *ill-conditioned*, then optimization almost certainly fails. Successful optimization relies on well-conditioned error gradients. This section studies the conditions of error gradients and weights' updates of the PlainNet, ResNet and ResNeXt during the backpropagation phase. For the analysis, we rely on the previous remarks for the different DNN models and the following lemmas, which are straightforward.

Lemma 4: Considering matrices $A \in \mathbb{R}^{m \times n}$ and $B \in \mathbb{R}^{n \times p}$. If B is singular, then the product, $C = AB$, is singular.

Lemma 5: Considering matrices $A \in \mathbb{R}^{m \times n}$ and $B \in \mathbb{R}^{n \times p}$. If A and B are non-singular, then the product, $C = AB$, is non-singular.

Definition 3: Considering the output of a hypothetical DNN layer m with linear activation function is $\boldsymbol{H}(\boldsymbol{x})^m$, the local error gradient at layer m, Δ^m, is given by

$$\Delta^m = \boldsymbol{H}(\boldsymbol{x})^m \left(W^{m+1}\Delta^{m+1}\right). \tag{34}$$

It is straightforward to obtain Definition 3 from the conventional backpropagation algorithm, given the hidden units employ the linear activation function.

Definition 4: Assuming the cost function of a DNN is C, the weight update at iteration t for a layer parameterized by W^m, and with input $\boldsymbol{H}(\boldsymbol{x})^{m-1}$ is

$$\Delta W^m(t) = -\eta\frac{\partial C}{\partial W^m} = \eta\Delta^m\boldsymbol{H}(\boldsymbol{x})^{m-1}, \tag{35}$$

where η is the learning rate.

1) PlainNet: The characteristics of the error gradient and weight updates during backpropagation for the PlainNet are discussed in the following remark.

Remark 4: The PlainNet's hidden representation $\boldsymbol{H}(\boldsymbol{x})^m_p$ for $m \longrightarrow \infty$ is singular from Remark 1, so it can be concluded using Lemma 4 and Definition 3 that the error gradient, Δ^m, is singular too. Similarly, the weight update, ΔW^m, in Definition 4 is singular. Hence, the singularity of error gradients and weight updates collaborate to plague optimization.

The consequence of Remark 4 is the loss of precision of computed results via numerical instability [63], [64]. Ultimately, the accumulation of precision errors can cause serious erratic weights updates, and thus optimization failure. This position is fascinating, considering that the study of error gradients of very deep PlainNet in [44] observed them as white noise (i.e. lacking structure). Interestingly, this finding is further validated by the results of our experiments.

2) ResNet and ResNeXt: Again, relying on previous remarks, lemma, corollary and definitions, we describe the characteristics of the error gradients and weights' updates of the ResNet and ResNeXt models are follows.

Remark 5: From Remark 2 and Corrollary 1, the ResNet's hidden representation $\boldsymbol{H}(\boldsymbol{x})^m_r$ and weight W^{m+1} for $m \longrightarrow \infty$ are both non-singular for the ResNet, respectively. Therefore, we can conclude from Lemma 5 and Definition 3 that the error gradient, Δ^m, is non-singular. Finally, from Lemma 5 and Definition 4, it is seen that ΔW^m is also non-singular.

Remark 5 shows that the error gradients in the ResNet are well-conditioned during training, as the hidden representations and model weights do not exhibit singularity.

Remark 6: We note for the ResNeXt the non-singularity of both the hidden layer representation $\boldsymbol{H}(\boldsymbol{x})^m_c$ and weight W^{m+1} for $m \longrightarrow \infty$ from Remark 3 and Corrollary 1, respectively. Subsequently, the error gradient, Δ^m, is non-singular given Lemma 5 and Definition 3. In addition, the non-singularity of ΔW^m reflects in Lemma 5 and Definition 4.

Overall, it is observed that both ResNet and ResNeXt circumnavigate singularity problems during backpropagation, and thus allow successful optimization.

V. Theoretical Analysis of Skip connections for Model Generalization

In this section, we build on the findings in Section IV for discussing how the learning characteristics of the PlainNet and models with skip connections, ResNet and ResNeXt, relate to model generalization. It is seen that the conditions of hidden layer representations not only contribute to the successful optimization of the models (as seen in Section IV), but also impact their generalization capacities. Namely, the relationship between the condition of the hidden representations and stability of solutions learned by the DNNs is given in the following proposition.

Proposition 2: Assuming that the already learned optimal solution for a DNN is θ. A relative change of the hidden representation at layer l, $\Delta\boldsymbol{H}(\boldsymbol{x})^l$, translates to a relative solution change, $\Delta\theta$, as follows

$$\frac{\parallel \Delta\theta \parallel}{\parallel \theta \parallel} \leq \kappa(\boldsymbol{H}(\boldsymbol{x})^l)\frac{\parallel \Delta\boldsymbol{H}(\boldsymbol{x})^l \parallel}{\parallel \boldsymbol{H}(\boldsymbol{x})^l \parallel} \;: 0 \leq l \leq m, \tag{36}$$

where $\theta = \{W^l\}_{l=1}^m$ and $\boldsymbol{H}(\boldsymbol{x})^0 = [\boldsymbol{x_1}, ..., \boldsymbol{x_N}]$ for $l = 0$ is the input to the DNN.

Proof. See Section A4 in the appendix for proof sketch.

Proposition 2 shows that the robustness of the already learned solution, θ, depends on $\kappa(\boldsymbol{H}(\boldsymbol{x})^l)$. That is, for good generalization, $\boldsymbol{H}(\boldsymbol{x})^l$ with small condition numbers are favoured. We note that $\Delta\boldsymbol{H}(\boldsymbol{x})^l$ is analogous to the change, which manifests when a trained DNN is tested with novel data. Our expectation is that the solution already learned by the DNN gives good performance on the novel data, given that the novel data comes from the same distribution as the data used for training the DNN. That is, the novel data for testing can be seen as only small changes in the individual data samples that constitute the training data.

A. PlainNet

The following remark summarizes the generalization behaviour of very deep PlainNets.

Remark 7: From Remark 1, the singularity of $\boldsymbol{H}(\boldsymbol{x})^m$ for $m \longrightarrow \infty$ for the PlainNet, translates to a theoretically infinite $\kappa(\boldsymbol{H}(\boldsymbol{x})^m)$ in (36). Hence, very small changes in the input data translates to extremely large changes in the solution. Importantly, the convergence of the PlainNet even on the training data, where data samples of the same class are slightly different is not guaranteed. Finally, model optimization almost certainly fails so that generalization is subsequently impossible.

The optimization failures for the PlainNet on different datasets, which corroborates Remark 7 are reported in the experiment section.

B. ResNet and ResNeXt

Remark 8: For the ResNet and ResNet, the non-singularity of the hidden representation $\boldsymbol{H}(\boldsymbol{x})^m$ for $m \longrightarrow \infty$ from Remark 2 and Remark 3 means a theoretically finite $\kappa(\boldsymbol{H}(\boldsymbol{x})^m)$ in (36). Therefore, for moderate values of $\kappa(\boldsymbol{H}(\boldsymbol{x})^m)$, we expect that small changes in the input data result in negligible changes in the solution.

Interestingly, experimental results show that both ResNet and ResNeXt operate with moderate condition numbers.

Furthermore, we study why the ResNeXt [34] mostly generalizes better than the ResNet [29]. For characterizing the subtle attribute of the ResNeXt that contributes to its improved generalization, we state the following theorem that relates the condition of a matrix to its dimension.

Theorem 5: (Rudelson-Vershynin [65]) Given a standard Gaussian random matrix, $B \in \mathbb{R}^{s\times z}$, with independent entries, the bounds on the distribution of its singular values is

$$\sqrt{z} - \sqrt{s} \leq \mathbb{E}\,\sigma_{min}(B) \leq \mathbb{E}\,\sigma_{max}(B) \leq \sqrt{z} + \sqrt{s}, \quad (37)$$

where $\sigma_{min}(B)$ and $\sigma_{max}(B)$ are the minimum and maximum singular values of B, respectively; $\mathbb{E}$ denotes expectation, and $s < z$.

Proof. See Theorem 2.6 in [65] for proof.

When $s = z$ so that the bound on $\sigma_{min}(B)$ in Theorem 5 becomes problematic, [65] further showed that the relation $\sigma_{min}(B) \sim \sqrt{s} - \sqrt{s-1}$ suffices. Importantly, considering that both ResNet and ResNeXt do not suffer optimization problems as seen in Remark 5 and Remark 6, the following corollary is particularly useful for characterizing the improved generalization capacity of the ResNeXt over the ResNet as observed in the literature [34].

Corollary 2: For an m-layer DNN, the condition numbers of the hidden representations at layer l in the ResNet and ResNeXt, $\boldsymbol{H}(\boldsymbol{x})_r^l$ and $\boldsymbol{H}(\boldsymbol{x})_c^l$, respectively, have the relation

$$\kappa(\boldsymbol{H}(\boldsymbol{x})_r^l) > \kappa(\boldsymbol{H}(\boldsymbol{x})_c^l) : 1 \leq l \leq m. \quad (38)$$

Proof. We can deduce $\boldsymbol{H}(\boldsymbol{x})_r^l \in \mathbb{R}^{n\times N}$ and $\boldsymbol{H}(\boldsymbol{x})_c^l \in \mathbb{R}^{q\times N}$ from Remark 2 and Remark 3 for the ResNet and ResNeXt, respectively. Furthermore, the discussion in Section III.B.3 shows that by construction $q < n$. Consequently, applying Theorem 5 to $\boldsymbol{H}(\boldsymbol{x})_r^l$ and $\boldsymbol{H}(\boldsymbol{x})_c^l$ with $q < n < N$ shows the following relations (i) $\mathbb{E}\,\sigma_{max}(\boldsymbol{H}(\boldsymbol{x})_r^l) > \mathbb{E}\,\sigma_{max}(\boldsymbol{H}(\boldsymbol{x})_c^l)$, and (ii) $\mathbb{E}\,\sigma_{min}(\boldsymbol{H}(\boldsymbol{x})_r^l) < \mathbb{E}\,\sigma_{min}(\boldsymbol{H}(\boldsymbol{x})_c^l)$. Finally, the proof concludes by using (11) for the condition numbers of $\boldsymbol{H}(\boldsymbol{x})_r^l$ and $\boldsymbol{H}(\boldsymbol{x})_c^l$.

Remark 9: Considering the ResNet and ResNeXt with $\boldsymbol{H}(\boldsymbol{x})_r^l \in \mathbb{R}^{n\times N}$ and $\boldsymbol{H}(\boldsymbol{x})_c^l \in \mathbb{R}^{q\times N}$, respectively, where $q < n$. Applying (38) to (36) shows that the ResNeXt exhibits a more stable solution than the ResNet.

In the section of experiments, it is seen that the the ResNeXt generalizes better than the ResNet, and the hidden representations of the ResNeXt have smaller condition numbers than the hidden representations of the ResNet.

VI. Experiments

A. Datasets and settings

In contrast to many analysis works [57], [52] on DNN that are mainly theoretical, we corroborate our theoretical analysis with extensive experiments using MNIST [66], CIFAR-10 [67], CIFAR-100 [67] and ImageNet-2012 [24] datasets. The experiments on CIFAR-10, CIFAR-100 and ImageNet datasets use the common data augmentation techniques as in [68]. In order to clearly demonstrate the training problems of very deep PlainNets, we train PlainNet, ResNet and ResNeXt having 164 layers on MNIST, CIFAR-10 and CIFAR-100 datasets. Due to the enormous computational requirement for the ImageNet-2012 dataset, PlainNet, ResNet [29] and ResNeXt [34] having 101-layers are trained. All the DNN models are CNNs that solely rely on several convolution layers and the softmax layer as the final (i.e. output) layer. Specifically, the trained model configurations follow the standard building blocks as proposed in the literature. The ResNet and ResNeXt follow the construction configurations in [29] and [34], respectively. For the PlainNets, the trained configurations are obtained by eliminating the skip connections from the corresponding ResNet configurations. The mini-batch gradient descent is used for optimization. Training hyperparameters include an initial learning rate of 0.1 that is annealed to a final value of 0.0001 during training, momentum rate of 0.9, weight decay of 10^{-4} and batch size of 128. All the models employ a maximum of 200 training epochs. Furthermore, all experiments use the rectified linear activation function. The experiments on the MNIST dataset use no data augmentation. The experiments on CIFAR-10, CIFAR-100 and ImageNet datasets use the standard data augmentation found in [32]. For our experiments, the configurations of the ResNet and ResNeXt models follow those in [29] and [34], respectively; the PlainNet models are obtained by simply eliminating the skip connections from the corresponding ResNet models that have been constructed.

Model	Train accuracy	Test accuracy	# of param.
PlainNet-164	11.38%	11.35%	2.5M
ResNet-164	99.58%	99.60%	2.6M
ResNeXt-164	100%	99.72%	2.5M

TABLE I: Results on MNIST dataset

Model	Train accuracy	Test accuracy	# of param.
PlainNet-164	10.26%	10.16%	2.5M
ResNet-164	99.93%	93.53%	2.6M
ResNeXt-164	99.96%	93.89%	2.5M

TABLE II: Results on CIFAR-10 dataset

Model	Train accuracy	Test accuracy	# of param.
PlainNet-164	1.00%	1.12%	2.5M
ResNet-164	99.83%	75.72%	2.6M
ResNeXt-164	99.94%	76.64%	2.5M

TABLE III: Results on CIFAR-100 dataset

Model	Train accuracy	Test accuracy	# of param.
PlainNet-101	15.37%	14.94%	44.1M
ResNet-101	88.56%	77.32%	44.5M
ResNeXt-101	89.74%	78.48%	44.3M

TABLE IV: Top-1 results on ImageNet dataset

B. Model Evaluations

For assessing the training characteristics of the different DNN models, we particularly focus on observing the following aspects (i) growth of units' activations with model depth (ii) weights updates in training (iii) conditions of the model weights with depth using the condition numbers obtained from singular values and (iv) conditions of hidden layer representations with model depth. Condition numbers are

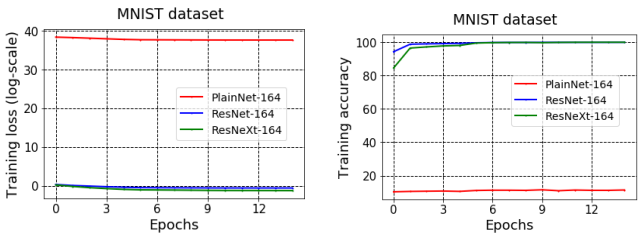

Fig. 3: Training curves for models on MNIST dataset. Left: Training loss. Right: Training accuracy

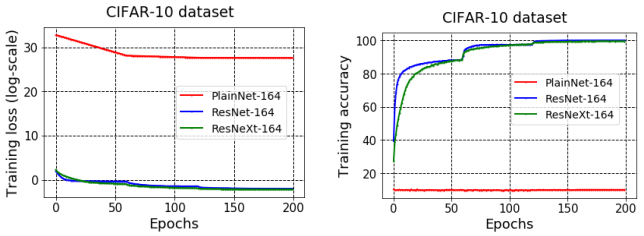

Fig. 4: Training curves for models on CIFAR-10 dataset. Left: Training loss. Right: Training accuracy

computed using equation (11). For the aforementioned aspects of interest, we focus on the early and later layers in the different models for investigation. Although our theoretical analyses use fully connected models where layer weights and hidden representations are both 2-dimensional arrays, it is straightforward to extend model evaluations to CNNs where layer weights and hidden representations are 4-dimensional and 3-dimensional tensors, respectively. Precisely, for obtaining the singular values of model tensors in the CNNs, we adopt the Higher Order SVD (HOSVD) [69] method that generalizes the SVD of matrices to tensors.

C. Results and discussion

Herein, the results of experiments are presented, along with discussions comparing and contrasting the different training properties of the DNNs.

Tables 1, 2, 3 & 4 show the obtained accuracies on the different datasets. It is noted that for all the datasets, the PlainNet models are clearly untrainable; there is an obvious optimization failure in learning, given the very poor accuracies on the training sets. For example, the PlainNet achieves 1% training accuracy on CIFAR-100 dataset; see Table 3. In contrast, the ResNet and ResNeXt exhibit no optimization problems, and on top of this generalize well on the datasets. Furthermore, the ResNeXt is seen to outperform ResNet on the datasets. This observation can be linked to our analysis, which is summarized in Remark 9.

The training curves for the different models on the MNIST and CIFAR-10 datasets are shown in Fig. 3 and Fig. 4, respectively. Note that the training losses on both dataset are plotted to log scale, as the PlainNet models have extremely high training losses, which reflect the severity of the optimization problem. The ResNet and ResNeXt models have significantly smaller training losses that show successful optimization.

The units' activations (i.e. outputs) and weights for the PlainNet are shown in Fig. 5 and Fig. 6, respectively. From Fig. 5, it is seen that units' activations in the first layer are extremely high; the range of units' activations is 1.7 million. This is very chaotic for optimization. At the one hundred and sixtieth layer (i.e. layer 160), units' activations have decreased to reasonable values. In Fig. 6, it is again seen that units' have extremely high weight values so that optimization is difficult; most weight values are between -200,000 and 200,000. We posit that the extremely high units' activations and weights values stem from the singularity of hidden representations and weights updates of the PlainNet discussed in Section IV.A.1 and Section IV.B.1. Units' activations and weights of the ResNet are given in Fig. 7 and Fig. 8, respectively. From Fig. 7, it is observed that the units operate with reasonable values in the first and one hundred and sixtieth layers; the range of units' activations is 6. Furthermore, units are seen to have reasonable weight values in the early and later layers, as in Fig. 8. These observations show why optimization is successful in the ResNet.

Furthermore, Fig. 9 and Fig. 10 show the units' activations and weights for the ResNeXt. Similar to the ResNet, it is seen in the early and later layers that the units activations and weights values are within reasonable ranges. Consequently, optimization is successful.

The condition number of the layer weights for the different models trained on MNIST and CIFAR-10 datasets are shown in Fig. 11. Fig. 12 shows the condition numbers of the layer weights using CIFAR-100 and ImageNet datasets. It is observed that the layer weights of the ResNet and ResNeXt have very small condition numbers, while the layer weights of the PlainNet have significantly higher condition numbers. These observations support theoretical results obtained in Section IV. In addition, to validate the theoretical results in Section IV and Section V, the condition numbers of the hidden representations of the different models on CIFAR-10 and CIFAR-100 datasets are reported in Fig. 13. The PlainNet trained on CIFAR-10 dataset, starting from the eightieth layer, have hidden representations with infinite condition numbers; on CIFAR-100 dataset, starting from the hundredth layer, the PlainNet's hidden representations have infinite condition numbers. This observation depicts the worst scenario of the singularity problem for optimization such that model generalization is impossible as given in Remark 8. In contrast, the hidden representations of the ResNet and ResNeXt never have infinite condition numbers; the condition numbers, which are high in the early layers quickly reduce to reasonable values so that optimization converges successfully. In addition, the hidden representations of the ResNeXt have smaller conditions than the ResNet. This observation aligns with our analysis that is summarized in Remark 9 for the improved generalization of the ResNeXt over ResNet.

Note that the MNIST dataset, which is generally considered an easy dataset to learn has been used for the analytical experiments, where the focus is observe the difficulty of optimizing very deep PlainNets; that is Fig.5 to Fig.10 . Otherwise, one might be curious as to if very deep PlainNets are trainable on easy datasets. Our results remove this curiosity, and directly suggest that the optimization conditions of units

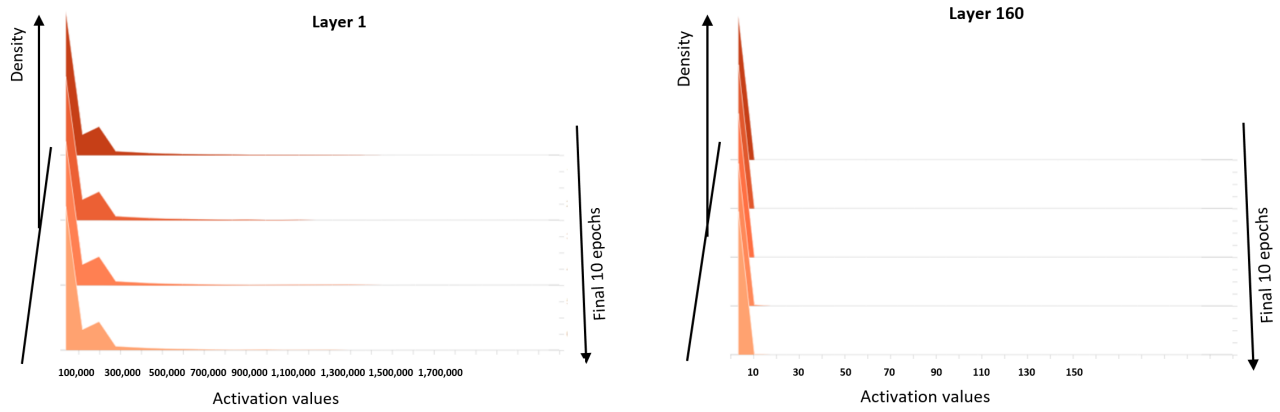

Fig. 5: PlainNet units' activations for 164 layer models trained on MNIST dataset

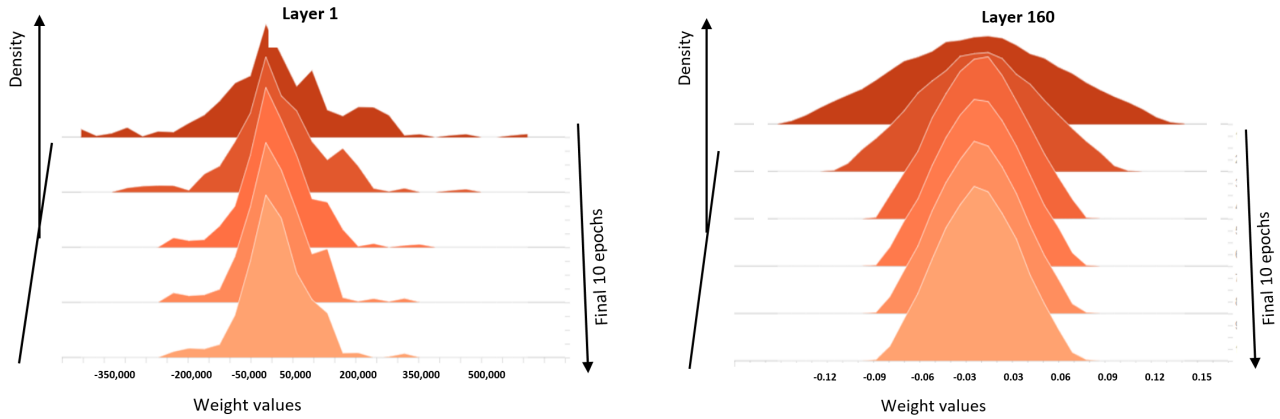

Fig. 6: PlainNet units' weights for 164 layer models trained on MNIST dataset

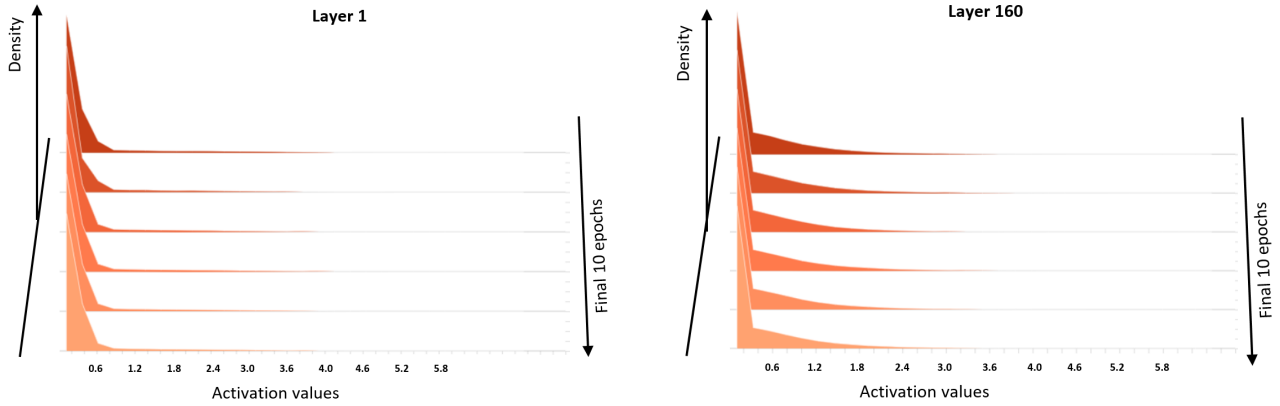

Fig. 7: ResNet units' activations for 164 layer models trained on MNIST dataset

activations (outputs), weights and gradients would be also chaotic, when training on harder datasets such as the CIFAR and Imagenet; this follows from the results in Tables I-IV. This position is reasonable since, it is well known that models, which cannot be properly optimized cannot generalize well. Contrary to expectation, our experiments clearly show that very deep PlainNets cannot be trained successfully even on the MNIST dataset. Additional results of the optimization conditions of PlainNet-164, ResNet-164 and ResNeXt-164 models trained on CIFAR-10 dataset in Table II are reported in Appendix A5 to further corroborate the positions given in this work. It will be seen that the optimization conditions of the different models trained on CIFAR-10 dataset that are discussed in Appendix A5 are similar to the same models trained on MNIST dataset.

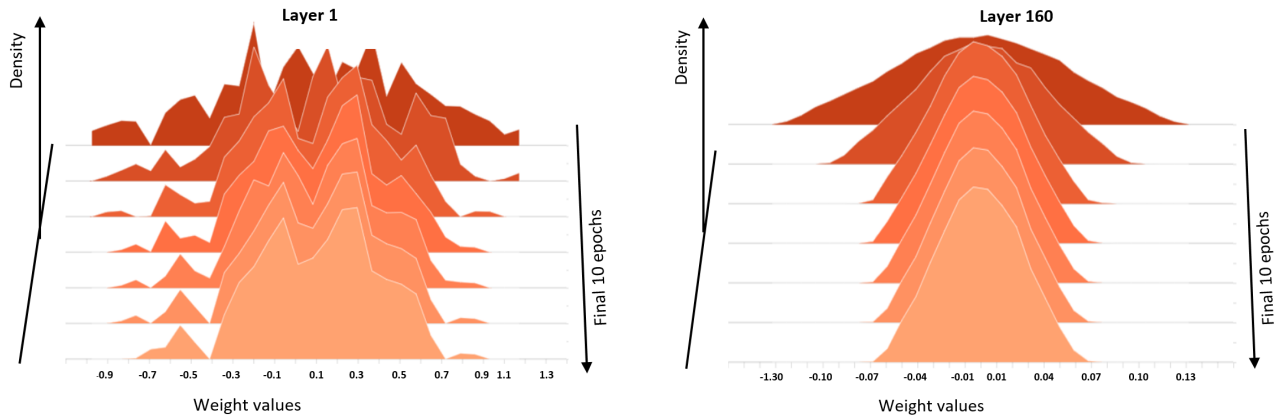

Fig. 8: ResNet units' weights for 164 layer models trained on MNIST dataset

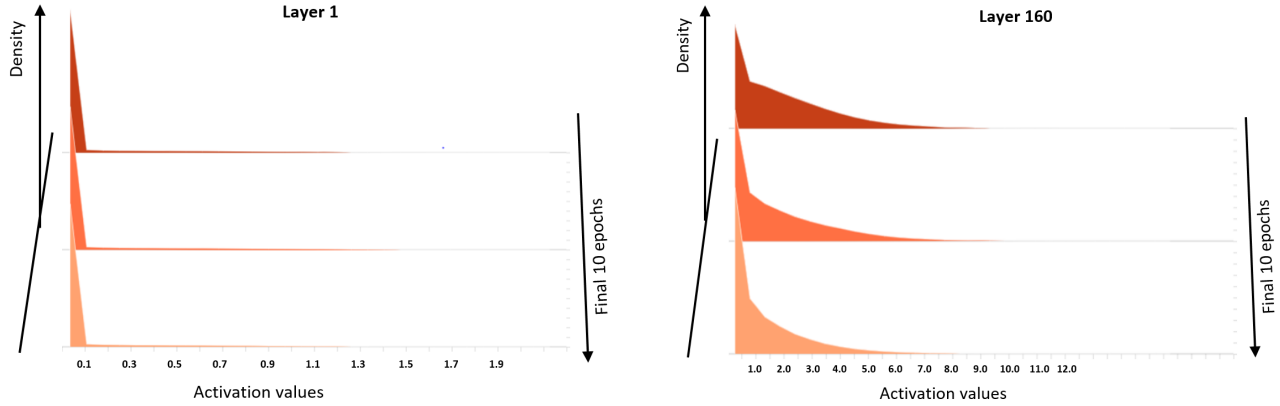

Fig. 9: ResNeXt units' activations for 164 layer models trained on MNIST dataset

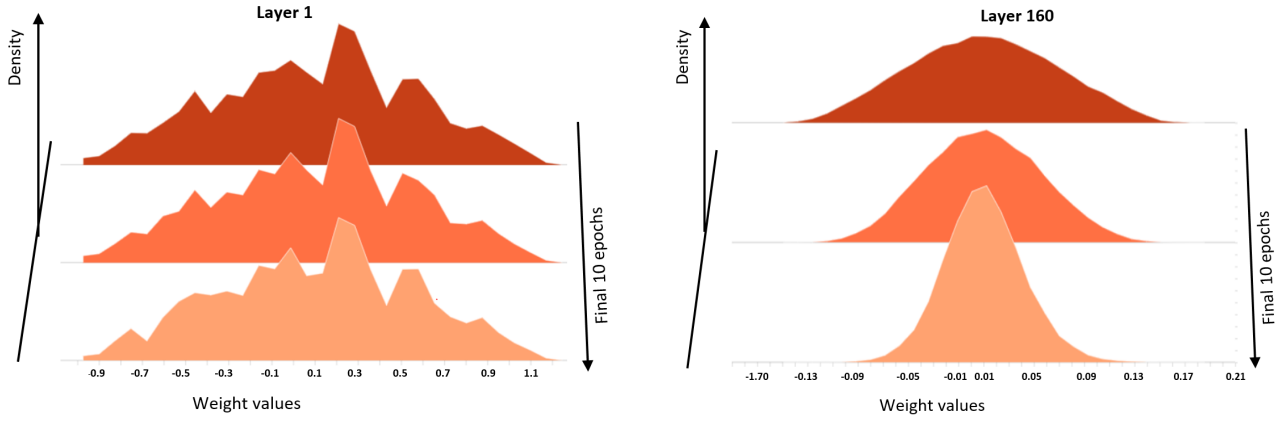

Fig. 10: ResNeXt units' weights for 164 layer models trained on MNIST dataset

VII. Training Schemes and similar Networks

This section discusses recent weight initialization schemes [70], [71] that claim to eliminate the need for skip connections for the succecssful training of very deep networks. Subsequently, we discuss the relationship between our results and the highway network [72]. Finally, we relate our findings with the results of Neural Architecture Search (NAS) with skip connections.

A. Weight Initialization

We refute the claims in [70], [71] that the special weight initializations suffice for training very deep models without skip connections. The results in [70], [71] does not match the state-of-the-art that are obtained from models with skip connections, and the proposal in [70] is even somewhat misleading.

1) Dirac weight initialization: the Dirac initialization in [70] for training very deep models is setup to imitate the

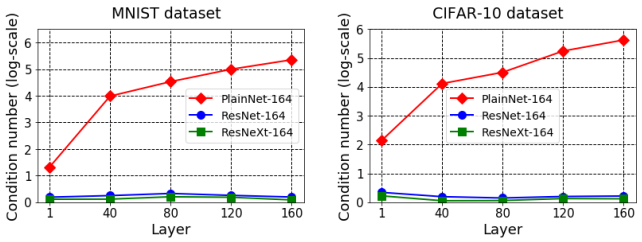

Fig. 11: Model weights condition number. Left: MNIST dataset. Right: CIFAR-10 dataset

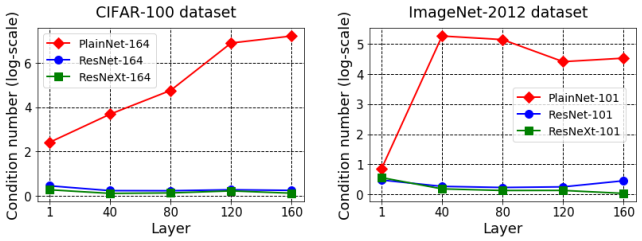

Fig. 12: Model weights condition number. Left: CIFAR-100 dataset. Right: ImageNet dataset

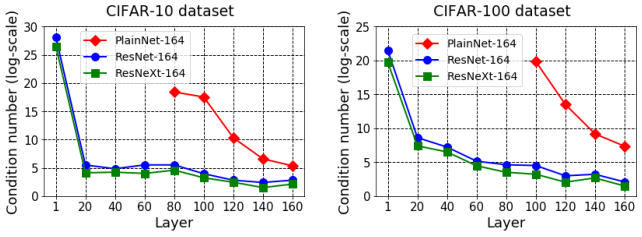

Fig. 13: Model hidden layer condition number; layers with missing condition numbers have infinite values. Left: CIFAR-10 dataset. Right: CIFAR-100 dataset

operation of the ResNet. Specifically, the model weights are reparameterized *at initialization* so that it behaves like it has skip connections as in

$$\widehat{\boldsymbol{W}}^l = \text{diag}(\boldsymbol{a})\mathbf{I} + \boldsymbol{W}^l, \tag{39}$$

where $\boldsymbol{W}$ is a random Gaussian or Uniform matrix; $\text{diag}(\boldsymbol{a})\mathbf{I}$ is the diagonal matrix with the vector $\boldsymbol{a}$ in the leading diagonal. Subsequently, the output of the Dirac model block, $\boldsymbol{h}(\boldsymbol{x})_d^l$, is

$$\boldsymbol{h}(\boldsymbol{x})_d^l = \widehat{\boldsymbol{W}}^l \widehat{\boldsymbol{W}}^{l-1} \boldsymbol{h}(\boldsymbol{x})_d^{l-2}. \tag{40}$$

It is stated in the work [70] that the 'the skip-connection in Dirac parameterization is implicit'; this is evident from (39) and (40), and dispels the conception that the Dirac initialization method has no basis in skip connections.

We observe that the initialization method shows some promise, as interesting results were obtained on small models, which are far from the state-of-the-art. However, the Dirac initialization results lagged the ResNet on large practical models. The problem that we observe with the Dirac parameterization is that the entries in $\boldsymbol{a}$ are free parameters so that $\text{diag}(\boldsymbol{a})\mathbf{I}$ can undesirably become a non-identity matrix during optimization. Consequently, the interesting properties of skip connections for training discussed in Sections IV & V are lost. In contrast, the identity matrices (from the explicit skip connections) in the parameterization of the ResNet and ResNeXt in (5) and (8), respectively, are always identity matrices (i.e. fixed) during training, so that the benefits of skip connections are always realized.

2) Delta-orthogonal weight initialization: the Delta-orthogonal initialization method in [71] is based on a mean field theory dynamical for information propagation via several model layers, so that dynamic isometry of the input-output Jacobian matrix is achieved. In this initialization method, the kernel weights are initialized as orthognal matrices with variance in the spatial center of the kernel and zero variance elsewhere. However, similar to the problem of Dirac initialization, this nice property may not persist during training so that optimization can become problematic. Although interesting results on the training accuracies were reported for DNNs with over 20 layers, the test accuracies obtained were so poor that the benefit of having several layers become questionable. For instance, the 32-layer and 128-layer models trained using the initialization method on CIFAR-10 dataset achieved uninspiring test accuracies of about 80% and 77%, respectively. We note that test accuracies of over 90% are readily obtained from much shallower models such as All-CNN [73], maxout networks [74] and Network-in-Network [75] models that all have less than ten layers. Interestingly, we find that the ResNet [29] with 110 layers achieved a good test accuracy of 93.57%.

B. Highway Network

We note that the analytical results align with the results of [72] in that the gating mechanism in the skip connections of the highway network allow some portion of the input data to be carried over to the different hidden layers so that the highway network operates similar to the ResNet. For instance, by setting the transform and carry gates $T(\boldsymbol{x}, \boldsymbol{W}_T)$ and $C(\boldsymbol{x}, \boldsymbol{W}_C)$, respectively both to the value of one in [72], the highway network exactly becomes the ResNet. As such, although our work did not directly analyze the highway network with a complicated architecture, our results provide an interesting basis for understanding the operation of the highway network as well.

C. Skip connections and neural architecture search

Recent works [76], [77] in NAS, where skip-connection is a candidate operator, have shown that the discovered DNNs are typically dominated by skip connections. Our results, which show that the incorporation of skip connections ensures a stable optimization for very deep neural network due to the non-singularity of the hidden representations can be used to explain why NAS has preference for many skip connections.

Subsequently, the model weights and gradients are well-behaved so that training is successful.

In fact, restriction on the number of skip connections that can be leveraged by NAS has been seen in the literature [77], [78]; this can be seen as a form of architecture regularization [77]. This is to enforce the participation of parameterized paths or operations during NAS. This architecture regularization is observed to improve the generalization performance of the discovered architectures [77], [78].

VIII. Main findings on skip connections

1) We find that very deep PlainNets suffer optimization problems due to the singularity of hidden representations that results from repeated multiplication of the input data with several layer weights. It is further seen that the singularity of the hidden representations reflects in the bad condition of the gradients and weight updates.
2) The ResNet and ResNeXt are easy to optimize because the skip connections eliminate the singularity of hidden representations. Subsequently, it is observed that the gradients and weight updates in the ResNet and ResNeXt are well conditioned so that training is successful.
3) We identify the better condition of the hidden representations of the ResNeXt as the reason why it usually generalizes better than the ResNet. By relating the size of random matrices and their conditions, we show that the smaller size of weights seen in the ResNeXt as compared to the ResNet is responsible for the improved generalization of the ResNeXt. To the best our knowledge, this observation is the first in the literature.
4) Our study of the special weight initialization techniques [70], [71] for alleviating the problem of training very deep PlainNets show that they considerably lag in performance when compared to models with skip connections. The main problem is that the weights can deviate from the desired operation regime during training so that optimization becomes problematic. A possible solution is to restrict the space of solution.

IX. Conclusion and future work

Astounding results on different learning tasks have been reported using very DNNs that employ skip connections. However, the optimization of very DNNs without skip connections referred to as PlainNet is very problematic; sometimes, very deep PlainNets are absolutely untrainable. Despite the extreme success of DNNs with skip connections, a concrete report of the distinct properties that allow their successful optimization and good generalization is lacking in the literature. We select two popular and very successful DNNs, ResNet and ResNeXt, as the focus of this study. Specifically, this paper investigates crucial model properties such as the singularity of hidden representations, error gradients and weights' updates, which are important for the successful optimization and generalization of DNNs. The provided analyses are confirmed by extensive experiments on four benchmarking datasets.

ResNet and ResNeXt models that both use the summation of preceding layer outputs with the current layer are studied in this paper. However, it would be interesting as a future work to study models that instead employ the concatenation of the preceding layer outputs with the current layer. A good example is the popular densely connected network (i.e. DenseNet) [36]. The DenseNet has been shown to perform extremely well on several tasks, and sometimes even outperforming the 'forerunner model', ResNet [29]. Surprsingly, little to no work has been carried out in understanding why the DenseNet that employs over 100 layers is trainable, and on top of that generalizes well.

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 and BRIDGES2020/IS/14755859/MEET-A/Aouada.

Appendix

A1. Proof of Proposition 1

Considering $W = \{\boldsymbol{w}_i\}_{i=1}^n$, where $\boldsymbol{w}_i$ is the i-th vector in W, and its elements are randomly sampled from a continuous distribution (i.e. uniform or Gaussian). Given $\boldsymbol{w}_i$, the matrix W excluding $\boldsymbol{w}_i$ is denoted W_{-i}; thus, the span of W_i is $W_{-i}^{span} = span(\boldsymbol{w}_1, .., \boldsymbol{w}_{n-1})$. Consequently, validating Proposition 1 translates to showing that any given $\boldsymbol{w}_i$ does not lie in the span of W_{-i}^{span}; that is, $\boldsymbol{w}_i$ and 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 [79]; thus, $P(w_1 \notin W_{-i}^{span}) = 1$. Furthermore, for $p \in \{2, ..., n-1\}$, let $W_{-1,p} = \{\boldsymbol{w}_1, ..., \boldsymbol{w}_p\}$ be the set of first p vectors, excluding vector i, such that $W_{-1,p}$ is linearly independent with a probability of 1. Therefore, we can state with a probablity of 1 that $W_{-1,p}$ 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. Proof of Lemma 2

First, let $W = BB^T$, and $B = \sum_{i=1}^n \sigma_i \boldsymbol{u_i} \boldsymbol{v_i}^T$ so that

$$W = \Big(\sum_{i=1}^{n} \sigma_i \boldsymbol{u_i} \boldsymbol{v_i}^T \Big) \Big(\sum_{j=1}^{n} \sigma_j \boldsymbol{v_j} \boldsymbol{u_j}^T \Big), \tag{41}$$

where an expansion gives

$$W = \sum_{i,j=1}^{n} \sigma_i \sigma_j \boldsymbol{u_i} (\boldsymbol{v_i}^T \boldsymbol{v_j}) \boldsymbol{u_j}^T. \tag{42}$$

Given, $\boldsymbol{v_i^T v_j} = 0$ for $i \neq j$, and $\boldsymbol{v_i^T v_i} = 1$, we have

$$W = \sum_{i}^{n} \sigma_i^2 \boldsymbol{u_i} \boldsymbol{u_i}^T. \tag{43}$$

From repeated multiplication, we can arrive at the general and compact expression

$$(W)^m = \sum_{i=1}^{n} \sigma_i^{2m} \boldsymbol{u_i} \boldsymbol{u_i}^T. \tag{44}$$

This concludes the proof. □

A3. Proof of Lemma 3

We rely on the method of characteristic functions [80] that sufficiently defines the probability distributions of real-valued random variables for this proof. Particularly, we leverage the fact that different random variables have distinct characteristic functions as supported by the *Inversion Formulae* [81], [82]. The characteristic function of an independent random variable X, $\varphi(u)$, is given as

$$\varphi_X(u) = \mathbb{E}(e^{iuX}), \tag{45}$$

where $i = \sqrt{-1}$ as in the imaginary unit, $u \in \mathbb{R}$ is a parameter of the function, and $\mathbb{E}$ denotes expectation. Furthermore, it can be shown that the characteristic function of a normally distributed random variable $X \sim \mathcal{N}(\mu, \beta^2)$ is

$$\varphi_X(u) = \exp\big(iu\mu_X - \beta_X^2 u^2/2\big). \tag{46}$$

Subsequently, the sum of the sequence of s independent random Gaussian variables $\{X_1 \sim \mathcal{N}(\mu_1, \beta_1^2), X_2 \sim \mathcal{N}(\mu_2, \beta_2^2), \cdots, X_k \sim \mathcal{N}(\mu_k, \beta_k^2), \cdots, X_s \sim \mathcal{N}(\mu_s, \beta_s^2)\}$, Y, can be expressed as the product of their respective characteristic functions as in

$$\varphi_Y(u) = \prod_{k=1}^{s} \exp\big(iu\mu_k - \beta_k^2 u^2/2\big), \tag{47}$$

$$\varphi_Y(u) = \exp\big(iu\sum_{k=1}^{s}\mu_k - \sum_{k=1}^{s}\beta_k^2 u^2/2\big). \tag{48}$$

The resulting characteristic function of Y shows that it is also a normally distributed variable as with any of its constituents $X_k \sim \mathcal{N}(\mu_k, \beta_k^2)$, but with mean $\sum_{k=1}^{s}\mu_k$ and variance $\sum_{k=1}^{s}\beta_k^2$. □

A4. Sketch of Proof for Proposition 2

First, let $\theta \in \mathbb{R}^{c\times n}$, $X \in \mathbb{R}^{n\times r}$ and $Y \in \mathbb{R}^{c\times r}$. Now, let us consider the simple problem, $Y = \theta X$: $X^\dagger$ is the pseudoinvrese of X. The objective is to estimate the solution, θ, given X and Y. Furthermore, let a *small* perturbation of ΔX result in a *small* solution perturbation, $\Delta\theta$, so that

$$Y = (\theta + \Delta\theta)(X + \Delta X). \tag{49}$$

Noting that $\Delta\theta\Delta X \approx 0$ and $Y = \theta X$, (49) becomes

$$\frac{\Delta\theta}{\theta} = -X^\dagger \Delta X. \tag{50}$$

Using Cauchy-Schwarz inequality for (50) yields

$$\frac{\parallel \Delta\theta \parallel}{\parallel \theta \parallel} \;\leq\; \parallel X^\dagger \parallel\parallel \Delta X \parallel \;\leq\; \parallel X^\dagger \parallel\parallel \Delta X \parallel \frac{\parallel X \parallel}{\parallel X \parallel}. \tag{51}$$

Finally, considering $\kappa(X) \approx \parallel X^\dagger \parallel\parallel X \parallel$, we obtain

$$\frac{\parallel \Delta\theta \parallel}{\parallel \theta \parallel} \;\leq\; \kappa(X)\frac{\parallel \Delta X \parallel}{\parallel X \parallel}. \;\square \tag{52}$$

A5. Additional Experiments using CIFAR-10 Dataset

This section presents additional experiments using CIFAR-10 dataset to further support the positions given in Section VIC of the paper. We train the PlainNet, ResNet and ResNeXt models with 164 layers on CIFAR-10 dataset. The units' activation (output) values and weights for the PlainNet are given in Fig. A1 and Fig. A2, respectively. It is seen in Fig. A1 that the units' outputs are extremely large for the first layer so that optimization is very unstable. Similarly, Fig. A2 shows that the units' weights for the first layer are well outside a reasonable range so that optimization is chaotic. As expected, the units' activations and weights in Fig. A1 and Fig. A2 are much larger than the PlainNet trained on MNIST dataset, which is a simpler dataset as shown in Fig. 5 and Fig. 6. This reflects a more chaotic optimization condition, since CIFAR-10 dataset is harder to learn than MNIST dataset; see Table I and Table II. Fig. A3 and Fig. A4 show the units' activations and weights for the ResNet, respectively. In contrast, it is observed that the units activations and weights have reasonable values so that optimization is not problematic. Finally, Fig. A5 and Fig. A6 show the the units' activations and weights for the ResNeXt, respectively. Again, the units' have activations and weights values that are reasonable for successful optimization.

References

[1] L. Nanni, S. Ghidoni, and S. Brahnam, "Handcrafted vs. non-handcrafted features for computer vision classification," *Pattern Recognition*, vol. 71, pp. 158–172, 2017.

[2] J. Zhang, Y. Xia, Y. Xie, M. Fulham, and D. D. Feng, "Classification of medical images in the biomedical literature by jointly using deep and handcrafted visual features," *IEEE journal of biomedical and health informatics*, vol. 22, no. 5, pp. 1521–1530, 2017.

[3] S. J. Oh, B. Schiele, and M. Fritz, "Towards reverse-engineering black-box neural networks," in *International Conference on Learning Representations*, 2018.

[4] M. Alber, S. Lapuschkin, P. Seegerer, M. Hagele, K. T. Schutt, G. Montavon, W. Samek, K.-R. Muller, S. Dahne, and P.-J. Kindermans, "innvestigate neural networks," *Journal of Machine Learning Research*, vol. 20, no. 93, pp. 1–8, 2019.

[5] A. Ghorbani, A. Abid, and J. Zou, "Interpretation of neural networks is fragile," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, 2019, pp. 3681–3688.

[6] S. S. Du, X. Zhai, B. Poczos, and A. Singh, "Gradient descent provably optimizes over-parameterized neural networks," in *International Conference on Learning Representations*, 2019.

[7] S. Du, J. Lee, H. Li, L. Wang, and X. Zhai, "Gradient descent finds global minima of deep neural networks," in *International Conference on Machine Learning*, 2019, pp. 1675–1685.

[8] P.-J. Kindermans, K. T. Schutt, M. Alber, K.-R. Muller, D. Erhan, B. Kim, and S. Dahne, "Learning how to explain neural networks: Patternnet and patternattribution," in *International Conference on Learning Representations*, 2018.

[9] W. Samek, A. Binder, G. Montavon, S. Lapuschkin, and K.-R. Müller, "Evaluating the visualization of what a deep neural network has learned," *IEEE transactions on neural networks and learning systems*, vol. 28, no. 11, pp. 2660–2673, 2016.

[10] L. M. Zintgraf, T. S. Cohen, T. Adel, and M. Welling, "Visualizing deep neural network decisions: Prediction difference analysis," 2017.

[11] F. Wang, H. Liu, and J. Cheng, "Visualizing deep neural network by alternately image blurring and deblurring," *Neural Networks*, vol. 97, pp. 162–172, 2018.

[12] Z. Wang, C. Li, X. Wang, and D. Wang, "Towards efficient convolutional neural networks through low-error filter saliency estimation," in *Pacific Rim International Conference on Artificial Intelligence*. Springer, 2019, pp. 255–267.

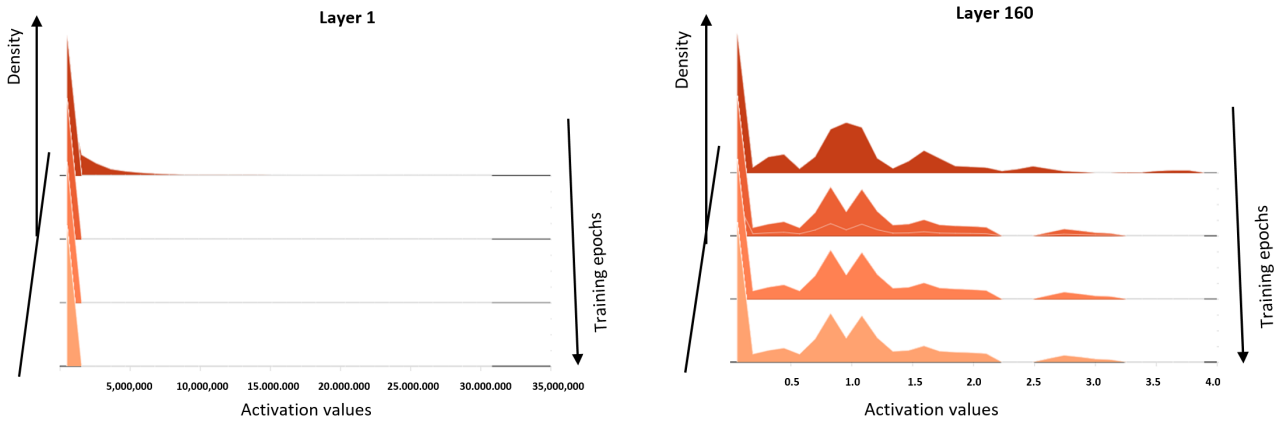

Fig. A1: PlainNet units' activations for 164 layer models trained on CIFAR-10 dataset

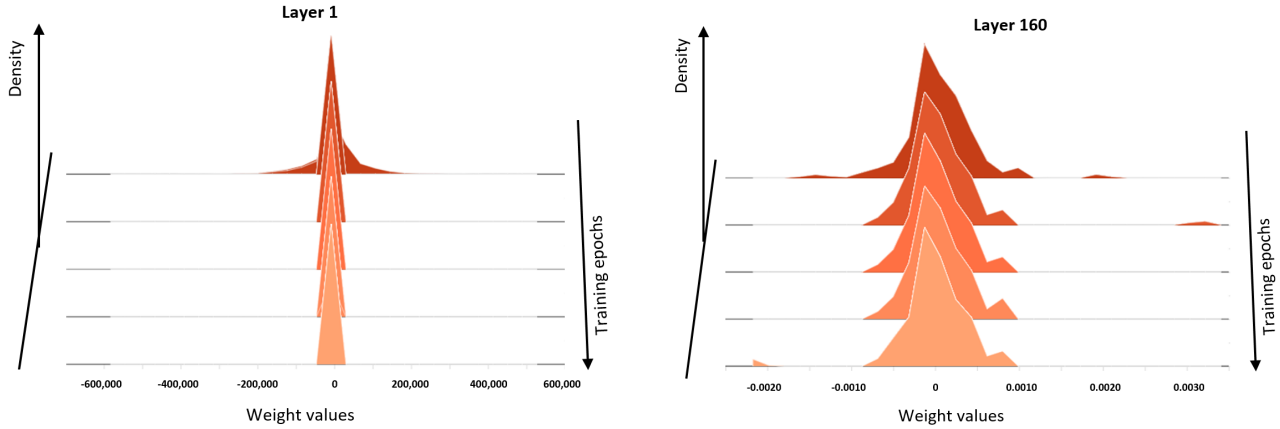

Fig. A2: PlainNet units' weights for 164 layer models trained on CIFAR-10 dataset

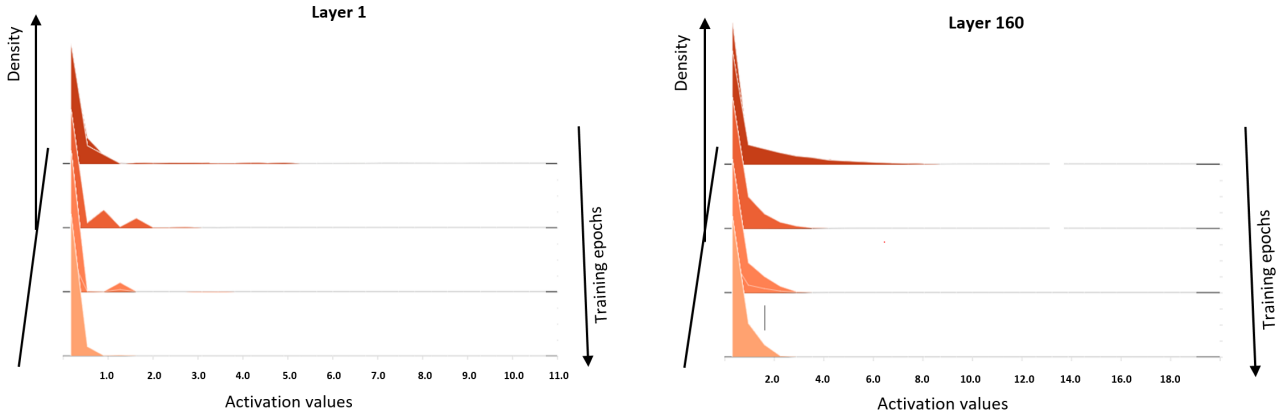

Fig. A3: ResNet units' activations for 164 layer models trained on CIFAR-10 dataset

[13] J. Zou, T. Rui, Y. Zhou, C. Yang, and S. Zhang, "Convolutional neural network simplification via feature map pruning," *Computers & Electrical Engineering*, vol. 70, pp. 950–958, 2018.

[14] P. L. Bartlett, N. Harvey, C. Liaw, and A. Mehrabian, "Nearly-tight vc-dimension and pseudodimension bounds for piecewise linear neural networks." *Journal of Machine Learning Research*, vol. 20, no. 63, pp. 1–17, 2019.

[15] H. Lin and S. Jegelka, "Resnet with one-neuron hidden layers is a universal approximator," in *Advances in neural information processing systems*, 2018, pp. 6169–6178.

[16] M. Raghu, B. Poole, J. Kleinberg, S. Ganguli, and J. S. Dickstein, "On the expressive power of deep neural networks," in *Proceedings of the 34th International Conference on Machine Learning-Volume 70*. JMLR. org, 2017, pp. 2847–2854.

[17] X. Cui, W. Zhang, Z. Tüske, and M. Picheny, "Evolutionary stochastic gradient descent for optimization of deep neural networks," in *Advances in neural information processing systems*, 2018, pp. 6048–6058.

[18] L. Wang, Y. Yang, R. Min, and S. Chakradhar, "Accelerating deep neural network training with inconsistent stochastic gradient descent," *Neural Networks*, vol. 93, pp. 219–229, 2017.

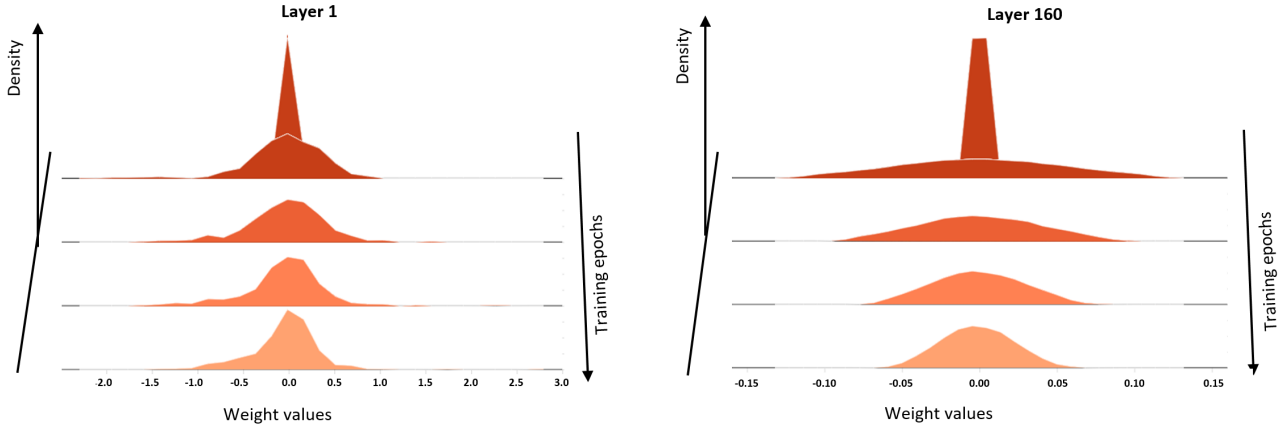

Fig. A4: ResNet units' weights for 164 layer models trained on CIFAR-10 dataset

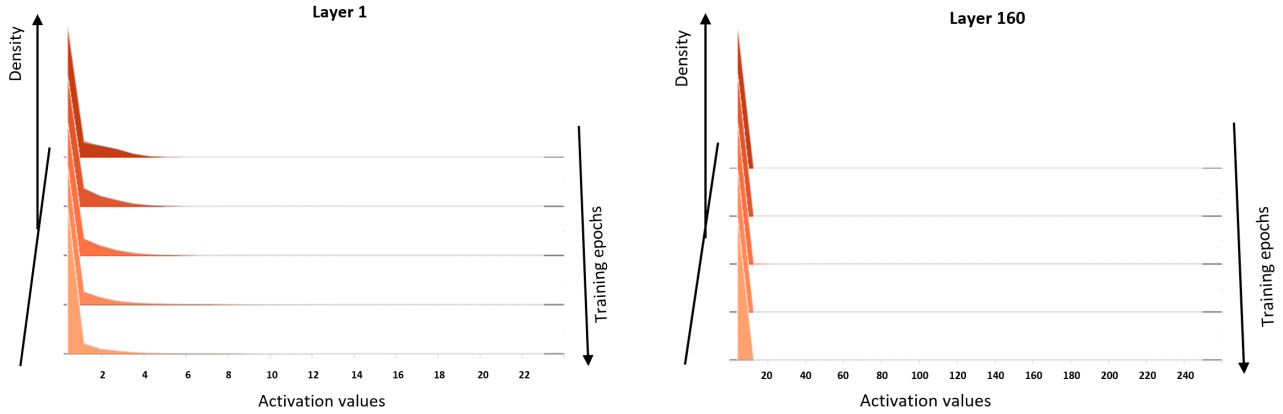

Fig. A5: ResNeXt units' activations for 164 layer models trained on CIFAR-10 dataset

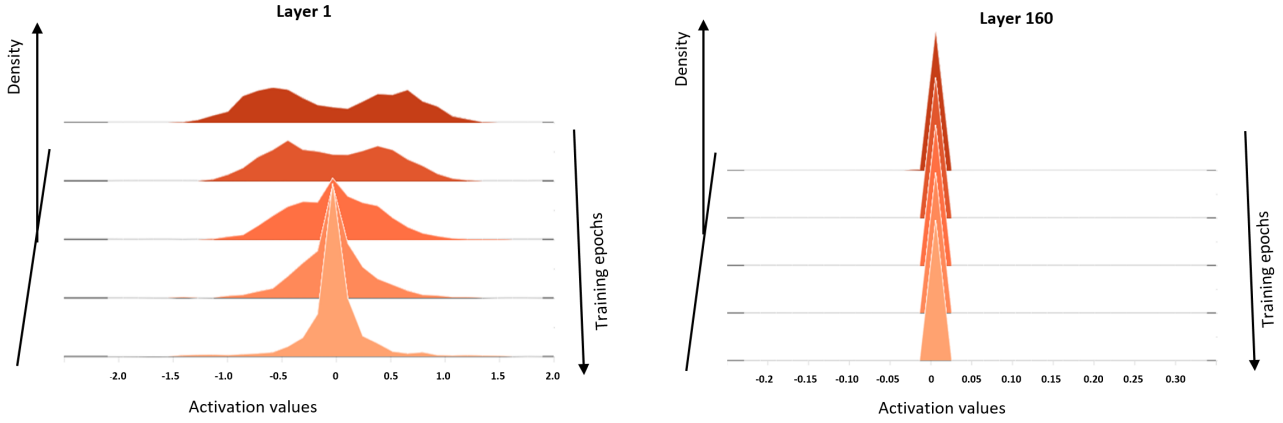

Fig. A6: ResNeXt units' weights for 164 layer models trained on CIFAR-10 dataset

[19] Z. Allen-Zhu, Y. Li, and Y. Liang, "Learning and generalization in overparameterized neural networks, going beyond two layers," in *Advances in neural information processing systems*, 2019, pp. 6155–6166.

[20] Y. Cao and Q. Gu, "Generalization bounds of stochastic gradient descent for wide and deep neural networks," in *Advances in Neural Information Processing Systems*, 2019, pp. 10 835–10 845.

[21] Y. Jiang, D. Krishnan, H. Mobahi, and S. Bengio, "Predicting the generalization gap in deep networks with margin distributions," in *International Conference on Learning Representations*, 2019.

[22] O. K. Oyedotun and A. Khashman, "Deep learning in vision-based static hand gesture recognition," *Neural Computing and Applications*, vol. 28, no. 12, pp. 3941–3951, 2017.

[23] D. CireşAn, U. Meier, J. Masci, and J. Schmidhuber, "Multi-column deep neural network for traffic sign classification," *Neural networks*, vol. 32, pp. 333–338, 2012.

[24] H. S. J. K. S. S. S. M. Z. H. A. K. A. K. M. B. A. C. B. Olga Russakovsky, Jia Deng* and L. Fei-Fei, "Imagenet large scale visual recognition challenge," *International Journal of Computer Vision (IJCV)*, pp. 211–252, December 2015.

[25] O. K. Oyedotun, A. El Rahman Shabayek, D. Aouada, and 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.

[26] M. Bianchini and F. Scarselli, "On the complexity of neural network classifiers: A comparison between shallow and deep architectures," *IEEE transactions on neural networks and learning systems*, vol. 25, no. 8, pp. 1553–1565, 2014.

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

[28] Z. Wu, C. Shen, and A. Van Den Hengel, "Wider or deeper: Revisiting the resnet model for visual recognition," *Pattern Recognition*, vol. 90, pp. 119–133, 2019.

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

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

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

[32] K. He, X. Zhang, S. Ren, and J. Sun, "Identity mappings in deep residual networks," in *European conference on computer vision*. Springer, 2016, pp. 630–645.

[33] G. Larsson, M. Maire, and G. Shakhnarovich, "Fractalnet: Ultra-deep neural networks without residuals," in *International Conference on Learning Representations*, 2017, pp. 1–11.

[34] S. Xie, R. Girshick, P. Dollár, Z. Tu, and 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.

[35] X. Zhang, Z. Li, C. Change Loy, and 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.

[36] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 4700–4708.

[37] C. Szegedy, S. Ioffe, V. Vanhoucke, and A. A. Alemi, "Inception-v4, inception-resnet and the impact of residual connections on learning," in *Thirty-First AAAI Conference on Artificial Intelligence*, 2017.

[38] O. K. Oyedotun, A. E. R. Shabayek, D. Aouada, and 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.

[39] K. He, X. Zhang, S. Ren, and 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.

[40] X. Glorot and 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.

[41] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *International Conference on Machine Learning*, 2015, pp. 448–456.

[42] K. Zhang, M. Sun, T. X. Han, X. Yuan, L. Guo, and T. Liu, "Residual networks of residual networks: Multilevel residual networks," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 28, no. 6, pp. 1303–1314, 2017.

[43] Y. Chen, J. Li, H. Xiao, X. Jin, S. Yan, and J. Feng, "Dual path networks," in *Advances in Neural Information Processing Systems*, 2017, pp. 4467–4475.

[44] D. Balduzzi, M. Frean, L. Leary, J. Lewis, K. W.-D. Ma, and 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.

[45] A. Veit, M. J. Wilber, and S. Belongie, "Residual networks behave like ensembles of relatively shallow networks," in *Advances in neural information processing systems*, 2016, pp. 550–558.

[46] K. Greff, R. K. Srivastava, and J. Schmidhuber, "Highway and residual networks learn unrolled iterative estimation," in *International Conference on Learning Representations*, 2017.

[47] S. Jastrzebski, D. Arpit, N. Ballas, V. Verma, T. Che, and Y. Bengio, "Residual connections encourage iterative inference," in *International Conference on Learning Representations*, 2018.

[48] E. Haber, L. Ruthotto, E. Holtham, and S.-H. Jun, "Learning across scales—multiscale methods for convolution neural networks," in *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[49] B. Chang, L. Meng, E. Haber, F. Tung, and D. Begert, "Multi-level residual networks from dynamical systems view," in *International Conference on Learning Representations*, 2018.

[50] E. Haber and L. Ruthotto, "Stable architectures for deep neural networks," *Inverse Problems*, vol. 34, no. 1, p. 014004, 2017.

[51] B. Chang, L. Meng, E. Haber, L. Ruthotto, D. Begert, and E. Holtham, "Reversible architectures for arbitrarily deep residual neural networks," in *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[52] Y. Zhou and Y. Liang, "Critical points of linear neural networks: Analytical forms and landscape properties," in *International Conference on Learning Representations*, 2019.

[53] S. Sonoda and N. Murata, "Transport analysis of infinitely deep neural network," *JMLR*, vol. 20, no. 1, pp. 31–82, 2019.

[54] A. M. Saxe, J. L. McClelland, and 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.

[55] K. Kawaguchi, "Deep learning without poor local minima," in *Advances in neural information processing systems*, 2016, pp. 586–594.

[56] Q. Nguyen and M. Hein, "Optimization landscape and expressivity of deep cnns," in *International Conference on Machine Learning*, 2018, pp. 3727–3736.

[57] T. Laurent and J. Brecht, "Deep linear networks with arbitrary loss: All local minima are global," in *International Conference on Machine Learning*, 2018, pp. 2902–2907.

[58] S. Sonoda and N. Murata, "Transport analysis of infinitely deep neural network," *The Journal of Machine Learning Research*, vol. 20, no. 1, pp. 31–82, 2019.

[59] N. Alexeev, F. Götze, and A. Tikhomirov, "On the singular spectrum of powers and products of random matrices," in *Doklady mathematics*, vol. 82, no. 1. SP MAIK Nauka/Interperiodica, 2010, pp. 505–507.

[60] C. Eckart and G. Young, "The approximation of one matrix by another of lower rank," *Psychometrika*, vol. 1, no. 3, pp. 211–218, 1936.

[61] J. W. Demmel, "The geometry of iii-conditioning," *Journal of Complexity*, vol. 3, no. 2, pp. 201–229, 1987.

[62] D. Duvenaud, O. Rippel, R. Adams, and Z. Ghahramani, "Avoiding pathologies in very deep networks," in *Artificial Intelligence and Statistics*, 2014, pp. 202–210.

[63] L. Elden, "Algorithms for the regularization of ill-conditioned least squares problems," *BIT Numerical Mathematics*, vol. 17, no. 2, pp. 134–145, 1977.

[64] S. Chen, "Local regularization assisted orthogonal least squares regression," *Neurocomputing*, vol. 69, no. 4-6, pp. 559–585, 2006.

[65] M. Rudelson and R. Vershynin, "Non-asymptotic theory of random matrices: extreme singular values," in *Proceedings of the International Congress of Mathematicians 2010 (ICM 2010) (In 4 Volumes) Vol. I: Plenary Lectures and Ceremonies Vols. II–IV: Invited Lectures*. World Scientific, 2010, pp. 1576–1602.

[66] Y. LeCun and C. Cortes, "Mnist handwritten digit database," http://yann.lecun.com/exdb/mnist/, Last accessed, October. 2019.

[67] A. Krizhevsky, V. Nair, and G. Hinton, "Cifar-10, cifar-100 (canadian institute for advanced research)," http://www.cs.toronto.edu/~kriz/cifar.html, Last accessed, October. 2019.

[68] S. Zagoruyko and N. Komodakis, "Wide residual networks," in *British Machine Vision Conference*, 2016, pp. 35–67.

[69] G. Bergqvist and E. G. Larsson, "The higher-order singular value decomposition: Theory and an application [lecture notes]," *IEEE Signal Processing Magazine*, vol. 27, no. 3, pp. 151–154, 2010.

[70] S. Zagoruyko and N. Komodakis, "Diracnets: Training very deep neural networks without skip-connections," *arXiv preprint arXiv:1706.00388*, 2017.

[71] L. Xiao, Y. Bahri, J. Sohl-Dickstein, S. Schoenholz, and J. Pennington, "Dynamical isometry and a mean field theory of cnns: How to train 10,000-layer vanilla convolutional neural networks," in *International Conference on Machine Learning*, 2018, pp. 5393–5402.

[72] R. K. Srivastava, K. Greff, and J. Schmidhuber, "Highway networks," *arXiv preprint arXiv:1505.00387*, 2015.

[73] J. T. Springenberg, A. Dosovitskiy, T. Brox, and M. Riedmiller, "Striving for simplicity: The all convolutional net," in *International Conference on Learning Representations Workshop*, 2015.

[74] I. Goodfellow, D. Warde-Farley, M. Mirza, A. Courville, and Y. Bengio, "Maxout networks," in *International conference on machine learning*, 2013, pp. 1319–1327.
[75] M. Lin, Q. Chen, and S. Yan, "Network in network," *arXiv preprint arXiv:1312.4400*, 2013.
[76] H. Liu, K. Simonyan, and Y. Yang, "Darts: Differentiable architecture search," in *International Conference on Learning Representations*, 2019.
[77] X. Chen, L. Xie, J. Wu, and Q. Tian, "Progressive differentiable architecture search: Bridging the depth gap between search and evaluation," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 1294–1303.
[78] T. E. Arber Zela, T. Saikia, Y. Marrakchi, T. Brox, and F. Hutter, "Understanding and robustifying differentiable architecture search," in *International Conference on Learning Representations*, 2020.
[79] J. M. Briggs and T. Schaffter, "Measure and cardinality," *The American Mathematical Monthly*, vol. 86, no. 10, pp. 852–855, 1979.
[80] R. B. Davies, "Numerical inversion of a characteristic function," *Biometrika*, vol. 60, no. 2, pp. 415–417, 1973.
[81] N. G. Shephard, "From characteristic function to distribution function: a simple framework for the theory," *Econometric theory*, vol. 7, no. 4, pp. 519–529, 1991.
[82] J. Gil-Pelaez, "Note on the inversion theorem," *Biometrika*, vol. 38, no. 3-4, pp. 481–482, 1951.

Oyebade Oyedotun is a Research Associate at the Interdisciplinary Centre for Security, Reliability, and Trust (SnT), University of Luxembourg since October 2020. He received the Ph.D. degree from the Computer Vision, Imaging and Machine Intelligence (CVI2) Research Group at the Interdisciplinary Centre for Security, Reliability and Trust (SnT), University of Luxembourg, L-1855 Luxembourg with a focus on deep learning, machine learning and vision applications. He received the M.Sc. degree in electrical and electronic engineering from Near East University, Lefkosa, N. Cyprus, in 2015. He has authored and co-authored many scientific articles in leading IEEE conferences and journals, including Transactions on Neural Networks and Learning Systems. He reviews for several journals, including IEEE TNNLS, IEEE TKDE, IEEE GRSL, IEEE Access, IEEE TAFFC, Neural Computing and Applications. His current research interests include machine learning and vision applications, neural networks, cognition modeling and neuroscience.

Dr. Kassem Al Ismaeil is a Research Associate at SnT. Kassem has received the BSc degree in electronic engineering and the MSc degree in computer science both from the University of Aleppo, Aleppo, Syria, in 2006 and 2008, respectively. He received the MSc degree in computer vision and robotics from the University of Burgundy, Le Creusot, France, in 2011, and the PhD degree in computer science from Interdisciplinary Centre for Security, Reliability, and Trust (SnT), University of Luxembourg, Luxembourg, in 2015. Later on, Kassem has joined the MunichRe group as an IT project manager where he was managing the middleware system implementation. His research interests include 3D computer vision, image and video processing, machine learning, with a focus on depth super-resolution, 3D reconstruction, and structure from motion.

Djamila Aouada is Senior Research Scientist and Assistant Professor at the Interdisciplinary Centre for Security, Reliability, and Trust (SnT), University of Luxembourg. She is Head of the Computer Vision, Imaging and Machine Intelligence (CVI2) Research Group at SnT, and Head of the SnT Computer Vision Laboratory. Dr. Aouada received the State Engineering degree in electronics in 2005, from the cole Nationale Polytechnique (ENP), Algiers, Algeria, and the Ph.D. degree in electrical engineering in 2009 from North Carolina State University (NCSU), Raleigh, NC, USA. Dr. Aouada has worked as a consultant for multiple renowned laboratories (Los Alamos National Laboratory, Alcatel Lucent Bell Labs., and Mitsubishi Electric Research Labs.). She has been leading the computer vision activities at SnT since 2009. Her research interests span the areas of image processing, computer vision, pattern recognition and data modelling. Dr. Aouada is Senior Member of the IEEE, member of the IEEE Signal Processing Society, IEEE WIE, INSTICC and the Eta Kappa Nu honor society (HKN). She has served as the Chair of the IEEE Benelux Women in Engineering Affinity Group from 2014 to 2016, Chair of the ECCV 2020 SHARP workshop, Area Chair at 3DV 2020, Chair of the CVPR 2021 SHARP workshop and Program Chair at 3DV 2021. She is the recipient of four IEEE best paper awards.