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 plain networks (PlainNets) that are absolutely untrainable beyond some depth. As such, the exposition of this article 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: 1) why DNNs with skip connections are easier to optimize and 2) 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, 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 article 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—Generalization, optimization, skip connection, very deep neural network (DNN).

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]–[5] and why simple gradient descent is generally able to find decent solutions for such highly nonconvex optimization problems [6], [7]. Nevertheless, considerable progress has been made in different aspects of understanding the operation of DNNs. Some of these include various visualization methods [8]–[11], saliency tracking and pruning for identifying important parameters [12], [13], analytical studies of model representational capacity [14]–[16], and works on model optimization [17], [18] and generalization [19]–[21].

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]–[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 plain networks (PlainNets), while both GoogleNet and residual network (ResNet) use some form of skip connections. Generally, training PlainNets with a few number of layers (i.e., typically one-to-ten layers) is not problematic. However, when model depth is extended beyond ten layers, training difficulty can ensue. This problem typically worsens with depth increase, and sometimes even the training set cannot be fit [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 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 ResNet [29], [32], FractaNet [33], residual network with aggregated features (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” fossters an incomplete knowledge; it is crucial to understand the unique characteristics of “model A” that result in performance improvement compared to “model B.”

In this article, 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 ResNeXt [34]. Our main contributions in this article are the following.

1) Analyze 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 article is organized as follows. Section II discusses the 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, this article 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 ten 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 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 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 autocorrelation function results, the gradients of the one-layer (i.e., shallow) PlainNet were found to resemble brown noise and thus allow successful optimization. In contrast, the gradients of the 24-layer (i.e., very deep) PlainNet were 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 fall between the brown noise and the white noise. As such, Baldazzi et al. [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 becomes more difficult as the gradients’ structure transits 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 less than their topological depths. Furthermore, Veit et al. [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, Veit et al. [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, Greff et al. [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 features in a specific stage. Subsequently, new hidden representations are computed at other stages in the ResNet. In the following work [47], it was observed that the ResNet mostly concentrates on representation learning in the ResNet block. In addition, considering the identity matrix as \(I\) in Fig. 2, we now obtain

\[
h(x)^{\frac{1}{2}} = W_{rb} \cdot h(x)^{-1} + h(x)^{-2}
\]

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

\[
h(x)^{\frac{1}{2}} = (W_{rb} W_{rb}^{-1} + I) h(x)^{-2}.
\]

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

\[
h(x)^{\frac{1}{2}} = (W_{rb} + I) h(x)^{-2}.
\]
3) Residual Network With Features Aggregation: The ResNeXt block [34] aggregates as output the summation of features learned via the different ResNeXt block as in Fig. 2, where \( W^{l,k}_g, W^{l-1,k}_g \in \mathbb{R}^{g \times q} \) with \( k = 1, \ldots, s \), 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 \( W^l_g \in \mathbb{R}^{n \times n} \) and \( W^l_g \in \mathbb{R}^{n \times n} \), respectively [see (2) and (5)]. In contrast, the ResNeXt block uses many smaller matrices for every transformation, as in \( W^{l,k}_g \in \mathbb{R}^{g \times q} \) given in Fig. 2. Note that the condition \( q < n \) will be important for analyzing the subtle difference between the ResNet and ResNeXt blocks in relation to generalization in Section V. For an input \( h(x)^{l-2} \in \mathbb{R}^n \), the output of the ResNeXt block, \( h(x)^l \in \mathbb{R}^q \), is

\[
h(x)^l = W^{l-1,k}_g W^{l-1,k}_g h(x)^{l-2} + \cdots + W^{l,k}_g W^{l-1,k}_g h(x)^{l-2} + \cdots + W^{l,k}_g W^{l-1,k}_g h(x)^{l-2} + h(x)^{l-2}. \tag{6}
\]

Similarly, for compactness, we let \( W^{l,k}_g = W^{l-1,k}_g \) in (6), where \( W^{l,k}_g \in \mathbb{R}^{g \times q} \). Finally, using \( I \in \mathbb{R}^{n \times n} \) and factorizing (6) gives

\[
h(x)^l = (W^{l,k}_g + \cdots + W^{l,k}_g + \cdots + W^{l,1}_g + I)h(x)^{l-2} \tag{7}
\]

which in a compact form is

\[
h(x)^l = \left( \sum_{k=1}^s W^{l,k}_g + 1 \right) h(x)^{l-2}. \tag{8}
\]

IV. THEORETICAL ANALYSIS OF SKIP CONNECTIONS FOR OPTIMIZATION

In this section, 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, and corollary.

Definition 1: If the vector set, \( \{w_i \in \mathbb{R}^n\}_{i=1}^n \), is the column of a nonsingular matrix \( W \in \mathbb{R}^{n \times n} \), then the vectors \( \{w_i\}_{i=1}^n \) are linearly independent and span \( \mathbb{R}^n \).

For the remaining parts of this article, 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 (See [58]): Given that 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 \( W \in \{W^1, \ldots, W^l, \ldots, W^m\} \) so that we have the products \( Y = \prod_{l=1}^m W^l \) and \( Z = (W)^m \), the asymptotic behavior of the singular spectrum of \( Y \) is the same as that of the singular spectrum of \( Z \).

Proof: See [58] for the proof that the limit distributions of the singular spectrum both \( Y \) and \( Z \) are the Fuss–Catalan distribution.

Proposition 1: Given a matrix \( W \in \mathbb{R}^{n \times n} \) whose column vectors, \( w_i \), are drawn from a uniform distribution or a Gaussian distribution, the probability that \( W \) is nonsingular, \( P(w_i \notin W^m) \), is

\[
P(W^m \notin W^m) = 1 : 1 \leq i \leq n. \tag{9}
\]

Proof: Section A1 in the supplementary material.

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 is all nonsingular; 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

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

where \( \sigma_i \) are the singular values of \( W \), \( u_i \in \mathbb{R}^n \) and \( v_i \in \mathbb{R}^n \) are the left and right singular vectors of \( W \), respectively, and \( T \) denotes the 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 \( \kappa \) of \( W \) given as

\[
\kappa(W) = \sigma_{\text{max}}(W)/\sigma_{\text{min}}(W) \tag{11}
\]
where $\sigma_{\text{max}}(W)$ and $\sigma_{\text{min}}(W)$ 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 [59], [60] specifically addresses this as follows.

**Theorem 1** (Eckart–Young [59], [60]): For a normalized nonsingular matrix $M$ (i.e., $\| M \| = 1$), its distance, $d$, to the set of ill-posed problems, $S$, is

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

**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

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

**Proof:** Section A2 in the supplementary material.

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

$$x = \sum_{i=1}^{n} \alpha_i u_i, \quad \alpha_i \in \mathbb{R}. \quad (14)$$

**Definition 2:** A collection of $N$ individual hidden layer representations at layer $l$, $h(x)^l \in \mathbb{R}^n$, compose the hidden representations data batch, $H(x)^l \in \mathbb{R}^{N \times n}$. Specifically, $h(x)^l = [h(x)^1, \ldots, h(x)^l, \ldots, h(x)^N]$.

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. In particular, 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:** Given the PlainNet in Section III-B1, we state the following theorem.

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

$$h(x)^m = \alpha_1 \sigma_p^{2m} u_1 \quad (15)$$

where $\sigma_p$ and $u_1$ are the first singular value component and first singular vector of $W_p^1$, respectively, and the scalar $\alpha_1$ is the first component of $\{\alpha_i\}_{i=1}^n$ as in (14).

**Proof:** The output of the PlainNet in the last layer $m$ can be written as

$$h(x)^m = \prod_{l=1}^{m} W_l^l x. \quad (16)$$

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

$$h(x)^m = (W_p)^m x. \quad (17)$$

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

$$h(x)^m = \sum_{i=1}^{n} \sigma_p^{2m} u_i u_i^T. x \quad (18)$$

where writing out (18) gives

$$h(x)^m = \left( \sigma_p^{2m} u_1 u_1^T + \sum_{i=2}^{n} \sigma_p^{2m} u_i u_i^T \right) x. \quad (19)$$

Subsequently, (19) is factorized so that we obtain

$$h(x)^m = \sigma_p^{2m} u_1 u_1^T x. \quad (20)$$

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

$$h(x)^m = \sigma_p^{2m} u_1 u_1^T x. \quad (21)$$

Putting (14) in (21) gives

$$h(x)^m = \sigma_p^{2m} u_1 u_1^T \sum_{i=1}^{n} \alpha_i u_i. \quad (22)$$

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

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

We note that using the deep Gaussian process along with some assumptions, Duvenaud et al. [61] arrived at a similar result as in Theorem 2. Namely, Duvenaud et al. [61] concluded that the representational capacity of DNNs collapses to a single degree of freedom as $m \rightarrow \infty$. In addition, the catastrophic performance 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 $\sigma_{p_i} = 0$ in Theorem 2, we obtain $h(x)^m = 0$ so that there is an absolute collapse of the PlainNet’s performance.

2) **Residual Network:** The next theorem relates the features conditioning of the ResNet as in Section III-B2.

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

$$h(x)^m = \alpha_1 u_1 \left[ (\sigma_r^2 + 1)^m - 1 \right] + \sum_{i=1}^{n} \alpha_i u_i \quad (23)$$

where $\sigma_r$ and $u_1$ are the first singular value component and first singular vector of $W_r^1$, respectively.

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

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

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

$$h(x)^m = (W_r + I)^m x. \quad (25)$$
Expanding \((W_r + I)^m\) in (25) using the binomial theorem, we have

\[
h(x)^m = (m(W_r)^{(m-1)}) + m(m - 1)(W_r)^{(m-2)}/2 + \cdots + 1)x. \tag{26}\]

Furthermore, using Lemma 2 for (26) gives

\[
h(x)^m = \left( \sum_{i=1}^{n} \sigma_i^{2m} u_i u_i^T + m \sum_{i=1}^{n} \sigma_i^{2m-1} u_i u_i^T + m(m-1) \right) \cdots \sum_{i=1}^{n} \sigma_i^{2m-1} u_i u_i^T / 2 + \cdots + 1 \right)x. \tag{27}\]

Given \(m \rightarrow \infty\) and \(\sigma_i > \sigma_j\), from Definition 2 and Theorem 3, the columns compactly written completes the proof of Theorem 3. □

In the worst case, where the deletion of all the bases \(\{u_i\}_{i=1}^{n}\) contributes to the computation of \(h(x)^m \in \mathbb{R}^q\) for \(m \rightarrow \infty\), repeated multiplication by \(\{W_l\}_{i=1}^{m}\) does not cause any basis to vanish. Hence, the ResNeXt retains full information. Again, using Definition 2 and Theorem 3, the columns of a data batch from the ResNet \(H(x)^m \in \mathbb{R}^{q \times N}\) are different, and therefore, there is no singularity.

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

**Remark 2**: Considering the input \(x\), all the bases \(\{u_i\}_{i=1}^{n}\) contribute to the computation of \(h(x)^m \in \mathbb{R}^q\) for \(m \rightarrow \infty\); repeated multiplication by \(\{W_l\}_{i=1}^{m}\) does not cause any basis to vanish. Subsequently, \(h(x)^m\) for the ResNet retains full information. From Definition 2 and Theorem 3, the columns of a data batch from the ResNet \(H(x)^m \in \mathbb{R}^{q \times N}\) are distinct, and thus, \(H(x)^m\) is nonsingular. 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 \(\sigma_i = 0\) in Theorem 3, we obtain \(h(x)^m = \sum_{i=1}^{q} a_i u_i\). Hence, \(h(x)^m = x\), and there is no catastrophic performance.

**3) ResNeXt**: The ResNeXt analysis is based on the discussion in Section III-B3 and will employ the following lemmas.

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

\[
X_s \sim \mathcal{N}\left(\sum_{k=1}^{s} \mu_k, \sum_{k=1}^{s} \beta_k^2\right). \tag{31}\]

**Proof**: Section A3 in the supplementary material. Namely, Lemma 3 allows us to replace \(\sum_{i=1}^{m} W_{l}^{i, k}\) in (8) with a single matrix \(W_l^i\). As such, we can express (8) compactly as

\[
h(x)^i = (W_l^i + 1)h(x)^{i-2} \tag{32}\]

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

**Theorem 4**: For the input \(x \in \mathbb{R}^n\) to a linear \(m\)-layer ResNeXt parameterized by \(\theta_c = \{W_l^i \in \mathbb{R}^{q \times q}, \ldots, W_l^i \in \mathbb{R}^{q \times q}, W_l^i \in \mathbb{R}^{q \times q}\}\), the hidden layer output, \(h(x)^m : m \rightarrow \infty\), is

\[
h(x)^m = a_1 u_1 \left( (\sigma_1^2 + 1) - 1 \right) \sum_{i=1}^{q} a_i u_i \tag{33}\]

where \(\sigma_i\) and \(u_i\) are the first singular value component and first singular vector of \(W_l^i\), 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 \(\{u_i\}_{i=1}^{q}\) contributes to the computation of \(h(x)^m \in \mathbb{R}^q\) for \(m \rightarrow \infty\), as repeated multiplication by \(\{W_l^i \}_{i=1}^{m}\) 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 \(H(x)^m \in \mathbb{R}^{q \times N}\) 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 circumvents catastrophic performance resulting from deleting \(m\) layer weights, that is, \(\sigma_i = 0\) in Theorem 4 so that \(h(x)^m = \sum_{i=1}^{q} a_i u_i\) and, thus, \(h(x)^m = x\).

**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 impacts 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 nonsingular, then the product, \(C = AB\), is nonsingular.
Definition 3: Considering the output of a hypothetical DNN layer $m$ with linear activation function is $\mathbf{H}(\mathbf{x})^m$, the local error gradient at layer $m$, $\Delta^m$, is given by

$$\Delta^m = \mathbf{H}(\mathbf{x})^m(W^{m+1} \Delta^{m+1}).$$

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

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

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

where $\eta$ 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 $\mathbf{H}(\mathbf{x})^m$ for $m \to \infty$ is singular from Remark 1, so it can be concluded using Lemma 4 and Definition 3 that the error gradient, $\Delta^m$, is singular too. Similarly, the weight update, $\Delta W^m$, in Definition 4 is singular. Hence, the singularity of error gradients and weight updates collaborates to plague optimization.

The consequence of Remark 4 is the loss of precision of computed results via numerical instability [62], [63]. Ultimately, the accumulation of precision errors can cause serious erratic weight 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 ResNetXt: Again, relying on previous remarks, lemma, corollary, and definitions, we describe the characteristics of the error gradients and weights’ updates of the ResNet and ResNetXt models are follows.

Remark 5: From Remark 2 and Corollary 1, the ResNet’s hidden representation $\mathbf{H}(\mathbf{x})^m$ and weight $W^{m+1}$ for $m \to \infty$ are both nonsingular for the ResNet, respectively. Therefore, we can conclude from Lemma 5 and Definition 3 that the error gradient, $\Delta^m$, is nonsingular. Finally, from Lemma 5 and Definition 4, it is seen that $\Delta W^m$ is also nonsingular.

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 ResNetXt the nonsingularity of both the hidden layer representation $\mathbf{H}(\mathbf{x})^m$ and weight $W^{m+1}$ for $m \to \infty$ from Remark 3 and Corollary 1, respectively. Subsequently, the error gradient, $\Delta^m$, is nonsingular given Lemma 5 and Definition 3. In addition, the nonsingularity of $\Delta W^m$ reflects in Lemma 5 and Definition 4.

Overall, it is observed that both ResNet and ResNetXt 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 ResNetXt, 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 $\theta$, a relative change of the hidden representation at layer $l$, $\Delta \mathbf{H}(\mathbf{x})^l$, translates to a relative solution change, $\Delta \theta$, as follows:

$$\frac{\| \Delta \theta \|}{\| \theta \|} \leq \kappa(\mathbf{H}(\mathbf{x})^l) \frac{\| \Delta \mathbf{H}(\mathbf{x})^l \|}{\| \mathbf{H}(\mathbf{x})^l \|} : 0 \leq l \leq m$$

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

Proof: Section A4 in the supplementary material. for proof sketch.

Proposition 2 shows that the robustness of the already learned solution, $\theta$, depends on $\kappa(\mathbf{H}(\mathbf{x})^l)$, that is, for good generalization, $\mathbf{H}(\mathbf{x})^l$ with small condition numbers are favored. We note that $\Delta \mathbf{H}(\mathbf{x})^l$ 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 come 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 behavior of very deep PlainNets.

Remark 7: From Remark 1, the singularity of $\mathbf{H}(\mathbf{x})^m$ for $m \to \infty$ for the PlainNet translates to a theoretically infinite $\kappa(\mathbf{H}(\mathbf{x})^m)$ in (36). Hence, very small changes in the input data translate 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 Section VI.

B. ResNet and ResNetXt

Remark 8: For the ResNet and ResNet, the nonsingularity of the hidden representation $\mathbf{H}(\mathbf{x})^m$ for $m \to \infty$ from Remarks 2 and 3 means a theoretically finite $\kappa(\mathbf{H}(\mathbf{x})^m)$
in (36). Therefore, for moderate values of $\kappa(H(x)^n)$, 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 [64]):** Given a standard Gaussian random matrix, $B \in \mathbb{R}^{x \times z}$, with independent entries, the bounds on the distribution of its singular values are

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

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

**Proof:** See [65, Th. 2.6] for proof.

When $s = z$ so that the bound on $\sigma_{\min}(B)$ in Theorem 5 becomes problematic, Rudelson and Vershynin [64] further showed that the relation $\sigma_{\min}(B) \sim \sqrt{x} - (s - 1)^{1/2}$ suffices. Importantly, considering that both ResNet and ResNeXt do not suffer optimization problems as seen in Remarks 5 and 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, $H(x)^l$ and $H(x)^l$, respectively, have the relation

$$\kappa(H(x)^l) > \kappa(H(x)^l) : 1 \leq l \leq m. \quad (38)$$

**Proof:** We can deduce $H(x)^l, x, H(x)^l \in \mathbb{R}^{q \times N}$ from Remarks 2 and 3, for the ResNet and ResNeXt, respectively. Furthermore, the discussion in Section III-B3 shows that by construction, $q < n$. Consequently, applying Theorem 5 to $H(x)^l$ and $H(x)^l$ with $q < n < N$ shows the following relations: 1) $\mathbb{E} \sigma_{\max}(H(x)^l) > \mathbb{E} \sigma_{\max}(H(x)^l)$ and 2) $\mathbb{E} \sigma_{\min}(H(x)^l) < \mathbb{E} \sigma_{\min}(H(x)^l)$. Finally, the proof concludes by using (11) for the condition numbers of $H(x)^l$, $H(x)^l$, and $H(x)^l$.

**Remark 9:** Consider the ResNet and ResNeXt with $H(x)^l, x, H(x)^l \in \mathbb{R}^{q \times N}$, 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 ResNeXt generalizes better than the ResNet, and the hidden representations of the ResNeXt have smaller condition numbers than those of the ResNet.

**VI. EXPERIMENTS**

**A. Datasets and Settings**

In contrast to many analysis works [52], [57] on DNN that are mainly theoretical, we corroborate our theoretical analysis with extensive experiments using MNIST [65], CIFAR-10 [66], CIFAR-100 [66], and ImageNet-2012 [24] datasets.

The experiments on CIFAR-10, CIFAR-100, and ImageNet datasets use the common data augmentation techniques as in [67]. In order to clearly demonstrate the training problems of very deep PlainNets, we train PlainNet, ResNet, and ResNeXt having 164 layers on the 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, a momentum rate of 0.9, a weight decay of $10^{-4}$, and a 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.

**B. Model Evaluations**

For assessing the training characteristics of the different DNN models, we particularly focus on observing the following aspects: 1) growth of units’ activations with model depth; 2) weights updates in training; 3) conditions of the model weights with depth using the condition numbers obtained from singular values; and 4) conditions of hidden layer representations with model depth. Condition numbers are computed using (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-D arrays, it is straightforward to extend model evaluations to CNNs where layer weights and hidden representations are 4-D and 3-D tensors, respectively. Precisely, for obtaining the singular values of model tensors in the CNNs, we adopt the higher order SVD (HOSVD) [68] method that generalizes the SVD of matrices to tensors.

**C. Results and Discussion**

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

Tables I–IV 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 the CIFAR-100 dataset (see Table III). 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 Figs. 3 and 4, respectively. Note that the training losses on both datasets are plotted to log scale, as the PlainNet models have extremely high training losses, which reflects 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 Figs. 5 and 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 160th 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 Sections IV-A1 and IV-B1.

Units’ activations and weights of the ResNet are given in Figs. 7 and 8, respectively. From Fig. 7, it is observed that the units operate with reasonable values in the 1st and 160th layers; the range of units’ activations is 6. Furthermore, units are seen to have reasonable weight values in the early and later layers, as shown in Fig. 8. These observations show why optimization is successful in the ResNet.

Furthermore, Figs. 9 and 10 show the units’ activations and weights for the ResNeXt, respectively. 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 is shown in Fig. 11. Fig. 12 shows the condition numbers of the layer weights using the 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 the theoretical results obtained in Section IV. In addition, to validate the theoretical results in Sections IV and V, the condition numbers of the hidden representations of the different models on the CIFAR-10 and CIFAR-100 datasets are reported in Fig. 13. The PlainNet trained on the CIFAR-10 dataset, starting from the 80th layer, has hidden representations with infinite condition numbers; on the 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

![Fig. 3. Training curves for models on the MNIST dataset. Left: training loss. Right: training accuracy.](image)

![Fig. 4. Training curves for models on the CIFAR-10 dataset. Left: training loss. Right: training accuracy.](image)
experiments, where the focus is to observe the difficulty of optimizing very deep PlainNets, that is, Figs. 5–10. 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’ activations (outputs), weights, and gradients would also be 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 the CIFAR-10 dataset in Table II are reported in Section A5 of the supplementary material to further corroborate the positions given in this work. It will be seen that the optimization conditions of the different models trained on the CIFAR-10 dataset that are discussed in Section A5 of the supplementary material are similar to the same models trained on the MNIST dataset.

VII. TRAINING SCHEMES AND SIMILAR NETWORKS

This section discusses recent weight initialization schemes [69], [70] that claim to eliminate the need for skip connections for the successful training of very deep networks. Subsequently, we discuss the relationship between our results and the HwNet [71]. 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] and [71] that the special weight initializations suffice for training very deep models without skip connections. The results in [70] and [71] do not match the state of the art that are obtained from models with skip connections, and the proposal in [69] is even somewhat misleading.

1) Dirac Weight Initialization: The Dirac initialization in [69] for training very deep models is setup to imitate the operation of the ResNet. Specifically, the model weights are reparameterized at initialization so that it behaves like that it has skip connections as in

\[
\hat{W}^l = \text{diag}(a)I + W^l
\]

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

\[
h(x)_{d^l} = \hat{W}^l \hat{W}^{l-1} h(x)_{d^{l-2}}.
\]

It is stated in the work [69] 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 initialization is that the entries in \( a \) are free parameters so that \( \text{diag}(a)I \) can undesirably become a nonidentity matrix during optimization. Consequently, the interesting properties of skip connections for training discussed in Sections IV and 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 [70] 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 orthogonal 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 becomes questionable. For instance, the 32- and 128-layer models trained using the initialization method on the CIFAR-10 dataset achieved uninspiring test accuracies of about 80% and 77%, respectively. We note that the test accuracies of over 90% are readily obtained from much shallower models such as All-CNN [72], maxout networks [73], and network-in-network [74] 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 mentioned in [71] in which the gating mechanism in the skip connections of the HwNet allows some portion of the input data to be carried over to the different hidden layers so that the HwNet operates similar to the ResNet. For instance, by setting the transform and carry gates \( T(x, W_T) \) and \( C(x, W_C) \), respectively, both to the value of one in [71], the HwNet exactly becomes the ResNet. As such, although our work did not directly analyze the HwNet with a complicated architecture, our results provide an interesting basis for understanding the operation of the HwNet as well.

C. Skip Connections and NAS

Recent works [75], [76] 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 DNN due to the nonsingularity 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 [76], [77]; this can be seen as a form of architecture regularization [76]. 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 [76], [77].

VIII. MAIN FINDINGS ON SKIP CONNECTIONS

The findings are listed in the following.

1) We find that very deep PlainNets suffer optimization problems due to the singularity of hidden representations that result 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 compared to the ResNet is responsible for the improved generalization of the ResNeXt. To the best of our knowledge, this observation is the first in the literature.

4) Our study of the special weight initialization techniques [69], [70] for alleviating the problem of training very deep PlainNets shows 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 article 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 article. 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 “fore-runner model,” ResNet [29]. Surprisingly, 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.

REFERENCES

[1] L. Nanni, S. Ghidoni, and S. Brahnam, “Handcrafted vs. non-handcrafted features for computer vision classification,” Pattern Recognit., vol. 71, pp. 158–172, Nov. 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 J. Biomed. Health Inform., vol. 22, no. 5, pp. 1521–1530, Sep. 2017.
[3] S. J. Oh, B. Schiele, and M. Fritz, “Towards reverse-engineering black-box neural networks,” in Proc. Int. Conf. Learn. Represent., 2018, pp. 1–20.
[4] M. Alber et al., “INNvestigate neural networks,” J. Mach. Learn. Res., vol. 20, no. 93, pp. 1–81, 2019.
[5] A. Ghorbani, A. Abid, and J. Zou, “Interpretation of neural networks is fragile,” in Proc. AAAI Conf. Artif. Intell., 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 Proc. Int. Conf. Learn. Represent., 2019, pp. 1–19.
[7] S. Du, J. Lee, H. Li, L. Wang, and X. Zhai, “Gradient descent finds global minima of deep neural networks,” in Proc. Int. Conf. Mach. Learn., 2019, pp. 1675–1685.
[8] P.-J. Kindermans et al., “Learning how to explain neural networks: Patternmat and patternattribution,” in Proc. Int. Conf. Learn. Represent., 2018, pp. 1–12.
[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 Trans. Neural Netw. Learn. Syst., vol. 26, no. 11, pp. 2660–2673, Nov. 2016.
[10] L. M. Zintgraf, T. S. Cohen, T. Adel, and M. Welling, “Visualizing deep neural network decisions: Prediction difference analysis,” in Proc. Int. Conf. Learn. Represent., 2017, pp. 1–12.
[11] F. Wang, H. Liu, and J. Cheng, “Visualizing deep neural network by alternately image blurring and deblurring,” Neural Netw., vol. 97, pp. 162–172, Jan. 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 Int. Conf. Artif. Intell., Cham, Switzerland: Springer, 2019, pp. 255–267.
[13] J. Zou, T. Rui, Y. Zhou, C. Yang, and S. Zhang, “Convolutional neural network simplification via feature map pruning,” Comput. Electr. Eng., vol. 70, pp. 950–958, Aug. 2018.
[14] P. L. Bartlett, N. Harvey, C. Liaw, and A. Mehrabian, “Nearly-tight VC-dimension and pseudodimension bounds for piecewise linear neural networks,” J. Mach. Learn. Res., vol. 20, no. 63, pp. 1–17, 2019.
[15] J. Wu, J. He, and Y. Todo, “The dendritic neuron model is a universal approximator,” in Proc. 6th Int. Conf. Syst. Informat. (ICSAI), Nov. 2019, 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 Proc. 34th Int. Conf. Mach. Learn., 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 Proc. Adv. Neural Inf. Process. Syst., 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 Netw., vol. 93, pp. 219–229, Sep. 2017.
[19] Z. Allen-Zhu, Y. Li, and Y. Liang, “Learning and generalization in overparameterized neural networks, going beyond two layers,” in Proc. Adv. Neural Inf. Process. Syst., 2019, pp. 6155–6166.
[20] Y. Cao and Q. Gu, “Generalization bounds of stochastic gradient descent for wide and deep neural networks,” in Proc. Adv. Neural Inf. Process. Syst., 2019, pp. 10835–10845.
[21] Y. Jiang, D. Krishnan, H. Mobahi, and S. Bengio, “Predicting the generalization gap in deep networks with margin distributions,” in Proc. Int. Conf. Learn. Represent., 2019, pp. 1–19.
[22] O. K. Oyedotun and A. Khashman, “Deep learning in vision-based static hand gesture recognition,” Neural Comput. Appl., vol. 28, no. 12, pp. 3941–3951, 2017.
[23] D. Ciresan, U. Meier, J. Masci, and J. Schmidhuber, “Multi-column deep neural network for traffic sign classification,” Neural Netw., vol. 32, pp. 333–338, Aug. 2012.
[24] O. Russakovsky et al., “ImageNet large scale visual recognition challenge,” Int. J. Comput. Vis. (IJCV), vol. 115, pp. 211–252, Dec. 2015.
[25] O. K. Oyedotun, A. E. R. Shabayek, D. Aouada, and B. Ottersten, “Highway network block with gates constraints for training very deep networks,” in Proc. IEEE/CVF Conf. Vis. Pattern Recognit. Workshops (CVPRW), Jun. 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 Trans. Neural Netw. Learn. Syst., vol. 25, no. 8, pp. 1553–1565, Aug. 2014.
[27] O. Delalleau and Y. Bengio, “Shallow vs. deep sum-product networks,” in Proc. Adv. Neural Inf. Process. Syst., 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 Recognit., vol. 90, pp. 119–133, Jun. 2019.
[29] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit., Jan. 2016, pp. 777–787.
[30] R. K. Srivastava, K. Greff, and J. Schmidhuber, “Training very deep networks,” in Proc. Adv. Neural Inf. Process. Syst., 2015, pp. 2377–2385.
[31] K. Simonyan and A. Zisserman, “Very deep convolutional networks for large-scale image recognition,” in Proc. Int. Conf. Learn. Represent., 2015, pp. 1–14.
[32] K. He, X. Zhang, S. Ren, and J. Sun, “Identity mappings in deep residual networks,” in Proc. Eur. Conf. Comput. Vis. Cham, Switzerland: Springer, 2016, pp. 630–645.
Oyebade K. Oyedotun (Student Member, IEEE) received the M.Sc. degree in electrical and electronic engineering from Near East University, Lefkosa, North Cyprus, in 2015, and the Ph.D. degree from the Interdisciplinary Centre for Security, Reliability, and Trust (SnT), Computer Vision, Imaging and Machine Intelligence (CVI2) Research Group, University of Luxembourg, Luxembourg City, Luxembourg, in 2020, with a focus on deep learning, machine learning, and vision applications.

He has been a Research Associate at the Interdisciplinary Centre for Security, Reliability, and Trust (SnT), University of Luxembourg, since October 2020. He has authored or coauthored many scientific articles in leading IEEE conferences and journals, including IEEE Transactions on Neural Networks and Learning Systems (TNNLS). He reviews several journals, including IEEE TNNLS, IEEE Transactions on Knowledge and Data Engineering, IEEE Geoscience and Remote Sensing Letters, IEEE Access, IEEE Transactions on Affective Computing, and Neural Computing and Applications. His current research interests include machine learning and vision applications, neural networks, cognition modeling, and neuroscience.

Kassem Al Ismaeil (Member, IEEE) received the B.Sc. degree in electronic engineering and the M.Sc. degree in computer science from the University of Aleppo, Aleppo, Syria, in 2006 and 2008, respectively, the M.Sc. degree in computer vision and robotics from the University of Burgundy, Le Creusot, France, in 2011, and the Ph.D. degree in computer science from the Interdisciplinary Centre for Security, Reliability, and Trust (SnT), University of Luxembourg, Luxembourg City, Luxembourg, in 2015.

Later on, he joined Munich Re, Munich, Germany, as an IT Project Manager, where he was managing the middleware system implementation. He is currently a Research Associate at SnT. His research interests include 3-D computer vision, image and video processing, and machine learning, with a focus on depth super-resolution, 3-D reconstruction, and structure from motion.

Djamila Aouada (Senior Member, IEEE) received the State Engineering degree in electronics from the École Nationale Polytechnique (ENP), Algiers, Algeria, in 2005, and the Ph.D. degree in electrical engineering from North Carolina State University (NCSU), Raleigh, NC, USA, in 2009.

She is currently a Senior Research Scientist and an Assistant Professor at the Interdisciplinary Centre for Security, Reliability, and Trust (SnT), University of Luxembourg, Luxembourg City, Luxembourg. She is also the Head of the Computer Vision, Imaging and Machine Intelligence (CVI2) Research Group, SnT, and the Head of the SnT Computer Vision Laboratory. She has worked as a consultant for multiple renowned laboratories (Los Alamos National Laboratory, Los Alamos, NM, USA; Alcatel Lucent Bell Labs, Murray Hill, NJ, USA; and Mitsubishi Electric Research Labs, Cambridge, MA, USA). She has been leading the computer vision activities at SnT since 2009. Her research interests include the areas of image processing, computer vision, pattern recognition, and data modeling.

Dr. Aouada is a member of the IEEE Signal Processing Society, IEEE WIE, INSTICC, and theEta Kappa Nu honor society (HKN). She was a recipient of four IEEE Best Paper Awards. She has served as the Chair for the IEEE Benelux Women in Engineering Affinity Group from 2014 to 2016, the European Conference on Computer Vision (ECCV) 2020 SHARP Workshop, and the International Conference on Computer Vision and Pattern Recognition (CVPR) 2021 SHARP Workshop, the Area Chair for International Conference on 3D Vision (3DV) 2020, and the Program Chair for 3DV 2021.