RESNETS ARE DEEPER THAN YOU THINK

Anonymous authors
Paper under double-blind review

ABSTRACT

The empirical success of residual connections is remarkable: nearly a decade after their introduction, they remain ubiquitous in modern neural network architectures. Their widespread adoption is usually attributed to improved trainability: residual networks train faster, more stably, and often achieve higher accuracy than their feedforward counterparts. While our experiments confirm the improved trainability of residual networks, we argue that this explanation is incomplete. We propose a complementary view: by combining shallow and deep paths in the computation graph of the network, residual connections can confer generalization benefits beyond pure optimization. Testing this hypothesis is surprisingly difficult: Simple side-by-side comparisons of residual and feedforward networks confound architectural prior with trainability, and prior attempts to close the gap do not isolate these effects conclusively. We therefore introduce a post-training partial-linearization setup that molds an already trained network into either a fixed-depth or a variable-depth architecture. In this setting, ordinary trainability differences are unlikely to explain the result. Across our experiments, variable-depth architectures consistently outperform fixed-depth counterparts at comparable nonlinear complexity, supporting the view that residual connections encode an inductive bias beyond numerical trainability.

1 INTRODUCTION

Combining deep neural networks and big data has enabled a series of dramatic breakthroughs; first in computer vision (Krizhevsky et al., 2012) but soon extending to a variety of different domains such as text processing (Llama Team, 2024), protein folding (Jumper et al., 2021), and multimodal tasks (Anil et al., 2023). The fact that this apparent success seems to extend to almost every type of natural data might be one of the most surprising results of modern deep learning research. In many cases, the inductive bias provided by network architectures and training protocols appears to be better aligned with natural data distributions than any other methods available today. A full understanding of the prior that the training of deep neural networks carries, especially wrt. singular network components, is considered an open research question to date.

Over the course of the last decade, thousands of different network architectures and design patterns were proposed. Out of those, very few stood the test of time, in the sense that they are still commonly used years after their introduction. Additionally, it seems to be the case that when swapping one network component for another, similar performances are often reached, making it difficult to draw conclusions regarding their potential hidden inductive biases. Notable exceptions to this rule include Transformer blocks (Vaswani et al., 2017), normalization layers (Ioffe and Szegedy, 2015), and residual connections (He et al., 2016), which are still to this day ubiquitous in current architectures. In this work, we focus on the latter, aiming to take a deeper look at the reasons for the immense success and perseverance of the residual architecture design since its introduction.

He et al. (2016) show that deep feedforward networks suffer from a degradation problem, where training performance deteriorates with depth, and propose residual connections as a *reformulation* that lets a layer learn the identity map more easily. The term reformulation, however, is ambiguous: it could mean either (i) a pure *reparametrisation* of the same function class, or (ii) the introduction of a genuinely different hypothesis space. Follow-up work investigated both interpretations, but the latter remains underexplored. Mean-field analysis by Yang and Schoenholz (2017a) traces the degradation to numerical instability: the authors show that well-conditioned *q- and c-maps*, which guarantee a forward/backward pass and non-vanishing rank, are highly predictive for network trainability and that

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

adding skip-connections mitigates these problems. Yet, Martens et al. (2021) and Zhang et al. (2022) show that even after one eliminates such numerical pathologies through carefully shaped initialization and re-scaling of the nonlinear layers, residual nets still outperform equally well-conditioned plain nets, suggesting additional factors at play.

A complementary perspective is given by Veit et al. (2016), who unravel a ResNet into an ensemble of exponentially many paths of different lengths and hypothesize that since deep paths result in very low gradient norms, they effectively might not contribute to the total gradient in any meaningful way. They conclude that residual connections do not magically make deep paths trainable; instead, they shorten the *effective path length* and thereby avoid exploding/vanishing gradients. However, despite being centered on the architectural shape of residual networks, their work only discusses the implications for trainability properties.

After extensive literature study, it appears clear that residual connections solve a number of crucial numerical issues in the optimization process; however, the following question remains unanswered:

> Does the *variable-depth prior* introduced by ResNets (i.e. the network containing both shallow and deep paths) help generalization performance beyond trainability?

We believe that answering this question conclusively bears important implications for both our understanding of the prior of deep learning, as well as the future design of neural network architectures. In this paper, we present our case in three arguments:

1. We argue that the above question cannot be answered conclusively by simple side-by-side training comparisons of residual and feedforward networks, since such comparisons confound architectural prior and trainability.
2. We give an analytical motivation for viewing residual connections as more than an equivalent reparametrization: under standard assumptions, a residual block cannot be represented by an equivalent plain block without expanding the intermediate representation.
3. We introduce a novel experimental setup where we extract comparable fixed-depth and variable-depth subnetworks from a fully trained network and show that the latter outperform the former.

2 Background and Motivation

2.1 Numerical Issues in Neural Network Training: Problems and Solutions

In this section, we review four commonly used criteria characterizing numerical problems during neural network training ("trainability issues" in the following) proposed by Lubana et al. (2021), and summarize the main classes of approaches proposed to mitigate them.

1. **Stable Forward Propagation.** The scale of a network's activations should not grow exponentially across layers during the forward pass. This effect was also sometimes called covariate shift (e.g. (Santurkar et al., 2018) Figure 2). Earlier network architectures, such as VGG (Simonyan and Zisserman, 2015) without normalization layers, had difficulties scaling beyond ~ 20 layers due to this issue.
2. **Non-Shattering Gradients / Stable Backward Propagation.** Gradients with respect to network inputs should maintain some degree of spatial auto-correlation / smoothness (ref. (Balduzzi et al., 2017), Fig. 1) for successful training with SGD, particularly with momentum. Ali Mehmeti-Göpel et al. (2021) show that the classical "exploding gradients" problem in deep feedforward networks (ref. (Yang et al., 2019b)) can be seen as a consequence of shattering gradients, as the gradients wrt. the network's weights also shatter, which in turn leads to high average gradient norms. Ali Mehmeti-Göpel and Wand (2024) (ref. Fig 5) and Schoenholz et al. (2017) show that exploding gradients can severely impede a network's trainability.
3. **Informative Forward Propagation:** The output rank, or more generally the singular value spectrum of the last layer's activations of a network, should not collapse. When the activations of the last layer become linearly dependent, the effective dimensionality

of the network function is reduced, and as a consequence, the network loses the ability to distinguish between different inputs and cannot learn effectively. This effect has been shown to occur for the product of many Gaussian matrices (ref. Saxe et al. (2014a) Figure 6) and also impedes a network's trainability (Schoenholz et al., 2017).

4. **Informative Backward Propagation** The rank (or singular value spectrum) of the network's input-output Jacobian should also not collapse, as this causes certain gradient directions to vanish, which in turn can impede trainability (Pennington et al., 2017).

In the following, we will review different approaches studied over the last decade to help mitigate the issues described above. The real difficulty lies in solving all of these problems at the same time, as a solution to a single problem can worsen the remaining ones, as we will see in the following.

Initialization Schemes Researchers first attempted to solve the first two issues by utilizing cleverly designed initialization schemes that guarantee a stable forward and backward propagation (Glorot and Bengio, 2010). Assuming that the effect of the nonlinear layers in the network is negligible, the authors find that for layers of different width, an initialization scheme can either satisfy condition 1 or 2 at the same time and derive an initialization scheme that averages both approaches as a compromise; however, this approach breaks down for deeper networks of non-constant width. He et al. (2015) later extended this idea to account for the effect of the nonlinear layer used on activations and gradients.

Dynamic Isometry / DKS There is a later line of research called *dynamic isometry*, which aims to initialize the network in a way that brings all singular values of the input-output Jacobian of the network function close to 1, consequently solving all three stability issues. For linear networks at initialization, it is possible to achieve dynamic isometry for networks with arbitrary depth using i.i.d. orthogonal matrices (Saxe et al., 2014b), but not for i.i.d. Gaussian matrices. As for nonlinear networks with sigmoid activations, a similar result can be found (Pennington et al., 2017), but the proof strategy involves shrinking the pre-activations into the linear regime of the activation, effectively linearizing the network. (Martens et al., 2021) recognize this problem and find a solution to create truly nonlinear deep feedforward networks that are also numerically stable.

Normalization Layers Normalization layers like Batch Normalization (Ioffe and Szegedy, 2015) ensure a stable forward pass at initialization and throughout training as per construction. However, as these layers usually do not perform a full whitening transformation, the covariance between channels can still vanish. Daneshmand et al. (2020) prove that BatchNorm retains at least the square root of the full rank at initialization for linear networks; however, their result does not hold for nonlinear networks or during training. There are attempts to implement full whitening normalization layers (Huang et al., 2019; 2018) that also normalize covariances between channels, but these add new failure modes such as stochastic axis swapping. Unfortunately, the combination of nonlinear layers and normalization layers also causes exploding gradients in the backward pass at initialization (Yang et al., 2019a; Luther, 2020).

Residual Connections The further addition of residual connections ensures an informative forward pass along with a stable backward pass Yang et al. (2019a). When properly scaling the residual branch, a stable forward pass can be achieved even without normalization layers (Zhang et al., 2019).

In practice, a combination of normalization layers and residual connections is used as a standard approach to guarantee a stable training dynamic.

2.2 Training Without Residual Connections

In this section, we discuss residual-free training approaches found in literature, evaluating whether an approach exists that solves all issues presented in Section 2 reliably during training.

Stable Backward Pass: Ali Mehmeti-Göpel and Wand (2024) show that it is possible to eliminate exploding/shattering gradients even in arbitrarily deep non-residual networks by simply using a very low initial learning rate that slowly increases in early training, i.e. a warm-up scheduler. Unlike approaches that rely on clever initialization schemes (e.g. Xiao et al. (2018); Balduzzi et al. (2017)), this property not only holds at initialization, but also reliably during training. The authors report a

remaining significant performance gap between deep feedforward and residual networks in Figure 5 of their work, even when completely eliminating exploding gradients using this technique.

Informative Forward/Backward Pass: As discussed in Section 2, full-whitening normalization-based approaches (ref. (Huang et al., 2019; 2018)) can reduce activation covariance anisotropy in the forward pass, but cannot recover zero-valued singular values. An analogous argument can be made for approximate second-order optimizers (e.g. K-FAC Martens and Grosse (2015)) in the backward pass. Moreover, such approaches carry numerical instabilities of their own, as explained below.

The strongest result attempting to "close the gap" that we are aware of is by Zhang et al. (2022), who report a remaining generalization gap of 1.4% on ImageNet for networks with 101 layers (ref. Table 9). However, the kernel shaping technique used by the authors does only guarantee numerical stability at initialization and relies on the K-FAC optimizer in a numerically highly unstable setting Yang and Schoenholz (2017b).

We argue that the reported performance gaps alone cannot be considered conclusive evidence that residual networks out-perform non-residual networks beyond training issues. The methods discussed above either introduce additional numerical complications, alter the effective nonlinear structure of the network, or fail to guarantee all stability conditions throughout training. Based on this evidence alone, the performance gap measured in literature cannot be attributed to the modified function space induced by residual connections; however, it can still be viewed as complementary evidence.

3 On Function Spaces

> "We present a residual learning framework to ease the training of networks that are substantially deeper than those used previously. We explicitly reformulate the layers as learning residual functions with reference to the layer inputs, instead of learning unreferenced functions."

This quote from the abstract of He et al. (2016) explicitly describes ResNets as a "reformulation" of a non-residual network function. As this terminology is ambiguous, this section's goal is to help the reader understand that ResNets are not merely a reparametrization of feedforward networks, but actually describe a different function space. We start by introducing precise definitions and notation.

Definition 1. *Let $f_\theta : \mathbb{R}^n \to \mathbb{R}^m$ be a network function with flattened input vector $x \in \mathbb{R}^n$ and weight vector $\theta \in \mathbb{R}^p$. We then define a **reparametrization** of the network function f as another network function $g_{\theta'} : \mathbb{R}^n \to \mathbb{R}^m$ with weight vector $\theta' \in \mathbb{R}^{p'}$ along with a weight reparametrization function $h : \mathbb{R}^p \to \mathbb{R}^{p'}$ such that:*

$$g_{h(\theta)}(x) = f_\theta(x).$$

*We call a reparametrization **equivalent**, if the networks f and g have the same width and depth.*

In the following, we establish that in the general case with a non-injective nonlinearity and square matrices, it is not possible to equivalently reparametrize a ResNet as a feedforward network.

Lemma 1. *Let $R : \mathbb{R}^n \to \mathbb{R}^n$ and $F : \mathbb{R}^n \to \mathbb{R}^n$ be a residual resp. feedforward block defined as follows:*

$$R(x) = \phi(\overline{W}x + \bar{b}) + x \tag{1}$$

$$F(x) = \phi(Wx + b), \tag{2}$$

where $W, \overline{W} \in \mathbb{R}^{n\times n}$ represent the weight matrices and $b, \bar{b} \in \mathbb{R}^n$ the biases and $\phi : \mathbb{R} \to \mathbb{R}$ an element-wise non-injective nonlinear function. We denote the set of all functions of these types as:

$$\mathcal{R}(x) := \{R(\overline{W}, \bar{b}) | \overline{W} \in \mathbb{R}^{n\times n}, \bar{b} \in \mathbb{R}^n\} \tag{3}$$

$$\mathcal{F}(x) := \{F(W, b) | W \in \mathbb{R}^{n\times n}, b \in \mathbb{R}^n\}. \tag{4}$$

Then the function spaces $\mathcal{F}$ and $\mathcal{R}$ differ, i.e. $\mathcal{F} \neq \mathcal{R}$.

Proof. For $\overline{W} = 0$, we obtain $R(x) = \phi(\bar{b}) + x$, which is injective. We now show that F is not injective, which implies that $I \notin \mathcal{F}$.

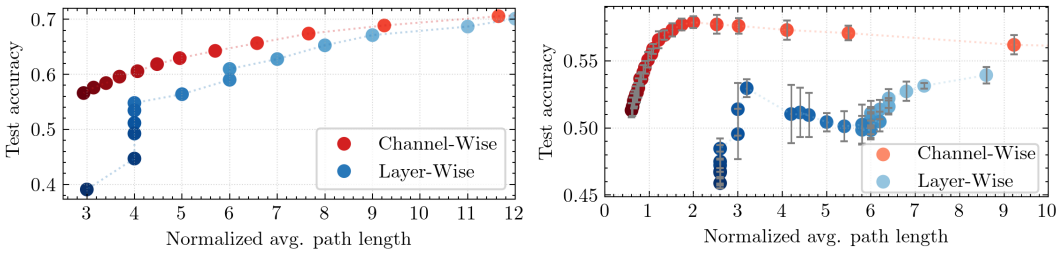


Figure 1: Comparing the test accuracies of partially linearized networks using a channel-wise or layer-wise partial linearization approach on the **ImageNet (left)** and **Cifar100 (right)** datasets. The results of the latter are averaged over 5 runs per data point, with standard deviation as error bars.

Case 1: W is not invertible. As W is square, F is therefore not injective.
Case 2: W is invertible. We assumed that ϕ is not injective and thus there exist scalars $c_1, c_2 \in \mathbb{R}$ with $c_1 \neq c_2$ such that $\phi(c_1) = \phi(c_2)$. Since W is invertible, we can find vectors $x_1, x_2 \in \mathbb{R}^n$ with $x_1 \neq x_2$ such that $\phi(Wx_1 + b) = \phi(Wx_2 + b)$ and thus F is not injective.

If every $F \in \mathcal{F}$ is not injective, whereas there exists $R \in \mathcal{R}$ which is injective, it follows that $\mathcal{R} \neq \mathcal{F}$. □

We now extend this result to entire networks, i.e. concatenation of such blocks.

Proposition 1. *We define the function space of residual resp. feedforward networks as:*

$$\hat{\mathcal{R}}(x) := \{R_1(\overline{W}_1, \bar{b}_1) \circ \ldots \circ R_\ell(\overline{W}_\ell, \bar{b}_\ell) | \overline{W}_i \in \mathbb{R}^{n \times n}, \bar{b}_i \in \mathbb{R}^n \quad \forall i = 1 \ldots \ell\} \tag{5}$$

$$\hat{\mathcal{F}}(x) := \{F_1(W_1, b_1) \circ \ldots \circ F_\ell(W_\ell, b_\ell) | W_i \in \mathbb{R}^{n \times n}, b_i \in \mathbb{R}^n \quad \forall i = 1 \ldots \ell\}. \tag{6}$$

Then the function spaces $\hat{\mathcal{F}}$ and $\hat{\mathcal{R}}$ differ, i.e. $\hat{\mathcal{F}} \neq \hat{\mathcal{R}}$.

Proof. If we apply Lemma 1 to the first layer, the proof follows immediately. □

We conclude that in the general case, assuming square matrices and non-injective nonlinear layers, the addition of residual connections cannot be considered a simple reparametrization: ResNets live in a different function space than plain feedforward networks.

In Appendix Section A.1, we discuss two different constructions that reparametrize ResNets as feedforward networks; they however require significant assumptions and double the width of the weight matrices. This does not contradict our point, as such constructions are possible, but not equivalent.

4 Empirical Evidence

The function-space argument above motivates the possibility that residual connections affect more than trainability. Whether this architectural prior improves generalization on real data is ultimately an empirical question. As we established in the previous section, numerical instability (i.e. an unstable forward/backward pass or vanishing rank in the last layer) greatly impedes the trainability of deep feedforward networks, so we need to exert great care in our experimental design to avoid such issues.

The basic idea is a setup very similar to Ali Mehmeti-Göpel and Disselhoff (2023); Dror et al. (2021): we start from an already pre-trained network and then mold it into different shapes in a post-training phase, in order to reduce the impact of trainability on the comparison. Starting from a deep pre-trained feedforward network, we gradually reduce the number of nonlinear units in the network using an additional regularization term that penalizes each nonlinear unit in the network during a post-training phase. Our goal is to mold the network either into a variable-depth network or a fixed-depth network during the post-training phase in order to compare their respective performances. We achieve this by using channel-wise, respectively layer-wise, linearization approaches in otherwise equal experimental settings: starting from a deep feedforward network, a channel-wise linearization can result in a variable-depth network (i.e. similar to a ResNet) if only a proportion of the channels

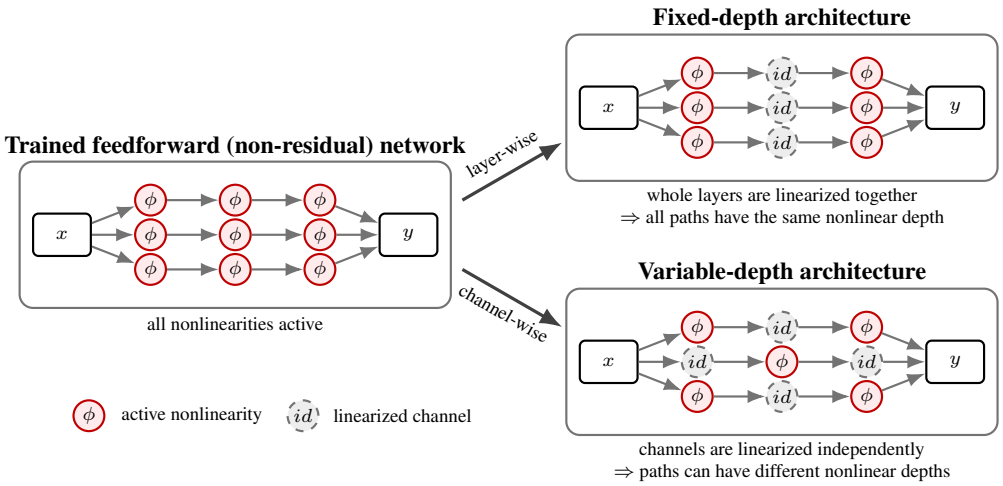


Figure 2: Illustration of how applying partial linearization to a trained feedforward network yields either a fixed-depth or variable-depth architecture. This technique allows us to transform a non-residual feedforward network into a variable-depth, "ResNet-like" computation graph *after* training.

within a layer become linear; a layer-wise linearization always results in a fixed-depth network (i.e. a shallower feedforward network). This concept is illustrated in Figure 2. Because both variants start from the same trained solution and differ only during the post-training linearization phase, this setup substantially reduces the trainability confound present in from-scratch comparisons. While the authors have done plenty of comparisons between their methods, the precise setup needed to make our claim (ref. Section 4) was never covered in the literature.

4.1 Glossary of Concepts

Partial linearization: In order to reduce the number of nonlinear units in a trained network, we replace its ReLU units with PReLU units (He et al., 2015) initialized with $\alpha_i = 0$ (i.e. reproducing a ReLU unit), and then add a sparsity regularization term

$$L_{0.5} = \sum |1 - \alpha_i|^{0.5} \tag{7}$$

to the regular training loss scaled with a *regularization weight* ω, where α_i are all layer/channel-wise PReLU slopes in the network. This way, the networks are incentivized to regularize some of their nonlinear units towards linearity, while preserving performance as much as possible. We set and freeze the slope parameter to $\alpha_i = 1$ if it gets close enough to one, i.e. $|\alpha_i - 1| < 0.01$. We call a nonlinearity inactive when $\alpha = 1$, as it reproduces a linear function, and active otherwise.

Layer-wise / channel-wise partial linearization: Partial linearization can be realized at two different levels of granularity: using a single slope parameter for each layer (layer-wise) or using a slope parameter for each individual channel (channel-wise).

Fixed-depth / variable-depth networks: When using the channel-wise technique, certain channels in a layer can become fully linear while others don't. The resulting network can therefore result in a variable-depth network, i.e. a network whose input-output function is composed of paths containing different numbers of nonlinear units. This corresponds to the definition of shallow and deep network paths in the view of Veit et al. (2016). When limiting the PReLU to one slope parameter per layer, the resulting network is merely composed of same-length paths, resulting in a fixed-depth network.

NAPL: Following Ali Mehmeti-Göpel and Disselhoff (2023) (ref. Section 3.1, pp. 3-4 for the exact definition), we measure the depth of partially linearized networks in average path length, which represents the average number of active nonlinear units encountered in a path from input to output through the computation graph of the network. Particularly, we use the width-agnostic normalized average path length (NAPL) as a measure, as otherwise wider layers dominate the measurement.

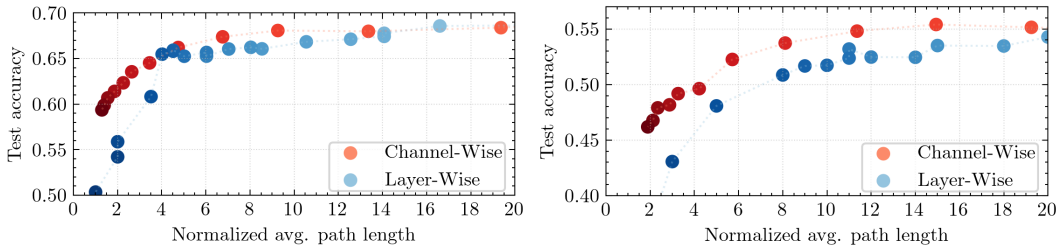


Figure 3: Comparing the test accuracies on Cifar100 of networks linearized using a channel-wise or layer-wise partial linearization method, starting from a **ResNet56 Short (left)** and **ResNet56 NoShort (right)**.

4.2 Experimental Setup

We chose RepVGG (Ding et al., 2021) as our starting architecture, as it is a recent network architecture without skip-connections but with competitive ImageNet performance, with a reasonable number of parameters, and publicly available weights. The RepVGG architecture does contain skip-connections across linear layers, which in our framework does not affect nonlinear depth as measured here by NAPL. This is intentional, as we want to show that nonlinear depth, which is only influenced by residual connections across nonlinear layers, is decisive for a network's expressivity. We chose specifically the RepVGG-A2 architecture with 23 layers, as it has a reasonably low parameter count but still achieves an ImageNet performance of 76.4%, similar to a ResNet50. We run our control experiment using standard ResNets with and without residual connections, to show that our results are not a specific quirk of the RepVGG architecture.

The post-training phase has a duration of 10/60 epochs on ImageNet resp. Cifar100. The longer post-training duration on the Cifar100 dataset shows that this effect does not vanish when post-training close to convergence. More implementation details can be found in the Appendix Section C. During post-training, we simply replace all ReLU units by channel-wise/layer-wise PReLU units as described in Section 4.1 and continue to train all parameters in the network.

4.3 Main Result: The Performance Gap

In Figure 1 (left), we show the test accuracies of the resulting partially linearized networks using a channel-wise and layer-wise technique starting from a pre-trained RepVGG-A2 model. We can clearly see that starting from networks with NAPL under 12, the layer-wise approach starts to be less performant than the channel-wise approach, and that the gap becomes bigger for shallower networks. We show similar results for Cifar10 in Appendix Section B.1. Interestingly, the NAPL where the performances of the two approaches diverge seems to be lower on easier datasets, which is coherent with our conjecture.

We also repeat this experiment on the Cifar100 dataset 5 times with different seeds from scratch, with a longer post-training phase, and report the results with error bars in Figure 1 (right) to highlight the robustness of this result. We see a slight increase of performance towards NAPL of around 3; this is possible as with increasing linearization the loss surface is also smoothened and therefore generalization performance can in fact slightly increase, before dropping for lower NAPL values as a result of low expressivity. Also on the Cifar100 dataset, we observe a significant performance gap between the layer-wise and channel-wise variants for lower NAPL, confirming the above result.

4.4 Control Experiment: Linearized ResNets

As a control setup, we design a similar experiment, but start our linearization process from a pre-trained residual network instead of a pre-trained feedforward network. If the effect we saw in Section 4 is greatly reduced, this implies that most of the observed benefit of the variable-depth path lengths can be replicated by regular residual connections, which in turn supports the interpretation that the performance gap of Section 4.3 can indeed be attributed to the different (variable vs fixed-depth) shape of the networks, instead of being a mere artifact of the channel-wise linearization procedure.

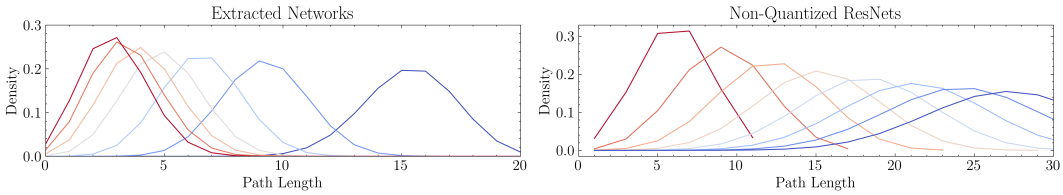


Figure 4: Comparing the histograms of networks extracted via **partial linearization (left)** versus **standard ResNets (right)**. The red color indicates a lower ω (left) or a lower depth (right).

For a setup with comparable residual and feedforward networks, we use a ResNet56 "Short" (i.e. with residual connections) and "NoShort" (i.e. without residual connections). When linearizing a ResNet in Figure 3 (left), we see that the layer-wise extracted networks only slightly underperform the channel-wise extracted networks for NAPL > 4. For NAPL 4 and below, we still see a bigger difference, which we conjecture is an artifact of very few free slope parameters being left in this case. The gap is significantly bigger when linearizing a feedforward network (right).

We also note that the networks that were extracted from a residual network (ref. Figure 3 left) generally outperform the networks extracted from a feedforward network (ref. Figure 3 right), supporting the view from the literature that residual networks generally train to a higher performance.

4.5 Sanity Checking the Shape of the Extracted Networks

We visualize the shape of the resulting partially linearized networks from Figure 3 (right) as a histogram of path lengths to verify that the networks extracted with the channel-wise method are indeed variable-depth shaped, as our claim is only plausible if this property holds. Note that this is not necessarily the case, as the resulting shape of the network is an emergent property of optimization. In Figure 4 (left), we confirm that the extracted networks are indeed variable-depth shaped, as they contain a mixture of shallow and deep paths, similar to a ResNet (Figure 4 right).

5 Discussion

In this work, we ask whether residual connections should be understood only as an optimization tool, or whether they also induce an architectural prior that affects generalization. The standard explanation for the success of residual networks is trainability: residual connections stabilize signal propagation, improve gradient flow, and allow much deeper networks to be optimized. While our results support this view, they also suggest that it is incomplete. In particular, they show that variable-depth networks (which include ResNets) preserve an already learned solution better than comparable fixed-depth networks (i.e. standard feedforward networks) when being pushed to the lower capacity end.

As a simple analytical motivation, we show that for square weight matrices and non-injective activations, residual blocks are not merely equivalent reparametrizations of plain feedforward blocks. This makes it plausible that residual connections do more than improve the conditioning of an otherwise equivalent parametrization: they change the architectural prior of the model.

Empirically testing the isolated effect of the differing architectural priors on generalization performance is difficult, as a direct from-scratch comparison between a residual network and a feedforward network changes at least two factors simultaneously: the optimization problem and the architectural prior. If the residual network performs better, the difference may be caused by improved trainability, by a more suitable function-space bias, or by an interaction between the two. Conversely, attempts to make deep feedforward networks trainable often introduce additional complications, such as altered nonlinear structure, new numerical instabilities, or stability guarantees that hold only at initialization. For this reason, while the remaining performance gaps reported in prior work do point in the direction of the architectural prior having an impact on generalization performance, they cannot by themselves cleanly identify whether residual connections provide an advantage beyond trainability.

Our post-training partial-linearization setup is designed to test a narrower question: does an already learned solution benefit more from a variable-depth prior than a fixed-depth one? Instead of training two different architectures from scratch, we start from the same trained feedforward parent network

and then mold it into either a fixed-depth or variable-depth shape. In the layer-wise case, all channels in a layer share a single slope parameter, so linearization removes nonlinearities layer by layer and the resulting network remains fixed-depth. In the channel-wise case, different channels can become linear independently, allowing the network function to form paths of different depths. By comparing the resulting networks at comparable nonlinear complexity, we test which computational structure better preserves an already learned solution.

Across our experiments, variable-depth networks preserve performance better than fixed-depth networks when nonlinear capacity is reduced and capacity is pushed towards the lower end. Since both networks are extracted from the same trained parent and differ only during the post-training linearization phase, the comparison substantially reduces the ordinary trainability confound present in from-scratch residual-vs-feedforward experiments. This shows that already learned solutions can indeed benefit more from a variable-depth prior than from a fixed-depth one, even in a setting where differences in trainability are unlikely to play a significant role.

The control experiment from Section 4.4 further supports this interpretation. When the same linearization procedure is applied to a network that already contains residual connections, the gap between channel-wise and layer-wise reductions mostly vanishes. This is the expected behavior if the advantage of the channel-wise extraction comes from its ability to recover a variable-depth structure instead of being a consequence of our specific experimental setup, such as the channel-wise method simply being easier to optimize or having marginally more slope parameters.

5.1 Limitations

In this Section, we discuss the limitations of our experimental results. First, in the experiment from Section 4, the partially linearized networks are extracted via gradient descent *after* training, which is necessary to limit the impact of the drastically different trainability properties of variable-depth and fixed-depth networks on our experiment. This does not reconstruct ordinary training dynamics; it tests preservation of learned solutions under controlled nonlinear-depth reductions. Also, since we are dealing with a non-convex optimization problem, which we optimize via a local gradient descent strategy, there is no guarantee that the networks extracted represent the exact sub-network with the best possible test accuracy given their target shape, i.e. that we reach a global minimum wrt. test accuracy in both the channel-wise and layer-wise cases. However, since merely the (relatively short) post-training phases differ between both approaches, and the network has already reached convergence before partial linearization, we argue that optimization is unlikely to make a difference in this comparison.

Another limitation of this experiment is that the partially linearized network using the channel-wise approach has slightly more free parameters compared to the layer-wise approach. To be exact, the parameter difference represents 7786 out of $\sim 26 \cdot 10^6$ parameters or $\sim 0.03\%$ of the total number. We address this concern in the Appendix Section B.3, where we present additional experiments that show that this small difference is highly unlikely to cause the performance gap observed.

6 Conclusion

We conclude that the long-standing performance gap between residual and feedforward networks should not be attributed only to numerical trainability. Residual connections also change the nonlinear path-length distribution of the computation graph, and this variable-depth structure appears to provide a useful inductive bias over learned solutions. This suggests that residual networks owe their success to both effects: they make deep networks easier to optimize, and they bias the represented computation toward mixtures of shallow and deep nonlinear paths which in turn can result in higher generalization performance.

AI Usage Statement and Reproducibility

The authors acknowledge the usage of LLM-based tools as a writing and polishing aid, for overall feedback, as well as for plot code writing. The authors take responsibility for the final manuscript. We further provide code along with launch scripts that allow the reader to fully reproduce all experiments described in this paper with a single command.

REFERENCES

Christian H. X. Ali Mehmeti-Göpel and Jan Disselhoff. Nonlinear advantage: Trained networks might not be as complex as you think. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett, editors, *International Conference on Machine Learning, ICML 2023, 23-29 July 2023, Honolulu, Hawaii, USA*, volume 202 of *Proceedings of Machine Learning Research*, pages 529–546. PMLR, 2023. URL https://proceedings.mlr.press/v202/ali-mehmeti-gopel23a.html.

Christian H. X. Ali Mehmeti-Göpel, David Hartmann, and Michael Wand. Ringing relus: Harmonic distortion analysis of nonlinear feedforward networks. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021. URL https://openreview.net/forum?id=TaYhv-q1Xit.

Christian H.X. Ali Mehmeti-Göpel and Michael Wand. On the weight dynamics of deep normalized networks. In *Forty-first International Conference on Machine Learning*, 2024. URL https://openreview.net/forum?id=AzUCfhJ9Bs.

Rohan Anil, Sebastian Borgeaud, Yonghui Wu, Jean-Baptiste Alayrac, Jiahui Yu, Radu Soricut, Johan Schalkwyk, Andrew M. Dai, Anja Hauth, Katie Millican, David Silver, Slav Petrov, Melvin Johnson, Ioannis Antonoglou, Julian Schrittwieser, Amelia Glaese, Jilin Chen, Emily Pitler, Timothy P. Lillicrap, Angeliki Lazaridou, Orhan Firat, James Molloy, Michael Isard, Paul Ronald Barham, Tom Hennigan, Benjamin Lee, Fabio Viola, Malcolm Reynolds, Yuanzhong Xu, Ryan Doherty, Eli Collins, Clemens Meyer, Eliza Rutherford, Erica Moreira, Kareem Ayoub, Megha Goel, George Tucker, Enrique Piqueras, Maxim Krikun, Iain Barr, Nikolay Savinov, Ivo Danihelka, Becca Roelofs, Anaïs White, Anders Andreassen, Tamara von Glehn, Lakshman Yagati, Mehran Kazemi, Lucas Gonzalez, Misha Khalman, Jakub Sygnowski, and et al. Gemini: A family of highly capable multimodal models. *CoRR*, abs/2312.11805, 2023. doi: 10.48550/ARXIV.2312.11805. URL https://doi.org/10.48550/arXiv.2312.11805.

David Balduzzi, Marcus Frean, Lennox Leary, J. P. Lewis, Kurt Wan-Duo Ma, and Brian McWilliams. The shattered gradients problem: If resnets are the answer, then what is the question? In Doina Precup and Yee Whye Teh, editors, *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, volume 70 of *Proceedings of Machine Learning Research*, pages 342–350. PMLR, 2017. URL http://proceedings.mlr.press/v70/balduzzi17b.html.

Hadi Daneshmand, Jonas Moritz Kohler, Francis R. Bach, Thomas Hofmann, and Aurélien Lucchi. Batch normalization provably avoids ranks collapse for randomly initialised deep networks. In Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin, editors, *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020. URL https://proceedings.neurips.cc/paper/2020/hash/d5ade38a2c9f6f073d69e1bc6b6e64c1-Abstract.html.

Xiaohan Ding, Xiangyu Zhang, Ningning Ma, Jungong Han, Guiguang Ding, and Jian Sun. Repvgg: Making vgg-style convnets great again. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2021, virtual, June 19-25, 2021*, pages 13733–13742. Computer Vision Foundation / IEEE, 2021. doi: 10.1109/CVPR46437.2021.01352. URL https://openaccess.thecvf.com/content/CVPR2021/html/Ding_RepVGG_Making_VGG-Style_ConvNets_Great_Again_CVPR_2021_paper.html.

Amir Ben Dror, Niv Zehngut, Avraham Raviv, Evgeny Artyomov, Ran Vitek, and Roy J. Jevnisek. Layer folding: Neural network depth reduction using activation linearization. *CoRR*, abs/2106.09309, 2021. URL https://arxiv.org/abs/2106.09309.

Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In Yee Whye Teh and D. Mike Titterington, editors, *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics, AISTATS 2010, Chia Laguna Resort, Sardinia, Italy, May 13-15, 2010*, volume 9 of *JMLR Proceedings*, pages 249–256. JMLR.org, 2010. URL http://proceedings.mlr.press/v9/glorot10a.html.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *2015 IEEE International Conference on Computer Vision, ICCV 2015, Santiago, Chile, December 7-13, 2015*, pages 1026–1034. IEEE Computer Society, 2015. doi: 10.1109/ICCV.2015.123. URL https://doi.org/10.1109/ICCV.2015.123.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*, pages 770–778. IEEE Computer Society, 2016. doi: 10.1109/CVPR.2016.90. URL https://doi.org/10.1109/CVPR.2016.90.

Kaixuan Huang, Yuqing Wang, Molei Tao, and Tuo Zhao. Why do deep residual networks generalize better than deep feedforward networks? - A neural tangent kernel perspective. In Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin, editors, *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020. URL https://proceedings.neurips.cc/paper/2020/hash/1c336b8080f82bcc2cd2499b4c57261d-Abstract.html.

Lei Huang, Dawei Yang, Bo Lang, and Jia Deng. Decorrelated batch normalization. In *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018*, pages 791–800. Computer Vision Foundation / IEEE Computer Society, 2018. doi: 10.1109/CVPR.2018.00089. URL http://openaccess.thecvf.com/content_cvpr_2018/html/Huang_Decorrelated_Batch_Normalization_CVPR_2018_paper.html.

Lei Huang, Yi Zhou, Fan Zhu, Li Liu, and Ling Shao. Iterative normalization: Beyond standardization towards efficient whitening. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16-20, 2019*, pages 4874–4883. Computer Vision Foundation / IEEE, 2019. doi: 10.1109/CVPR.2019.00501. URL http://openaccess.thecvf.com/content_CVPR_2019/html/Huang_Iterative_Normalization_Beyond_Standardization_Towards_Efficient_Whitening_CVPR_2019_paper.html.

Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In Francis R. Bach and David M. Blei, editors, *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015*, volume 37 of *JMLR Workshop and Conference Proceedings*, pages 448–456. JMLR.org, 2015. URL http://proceedings.mlr.press/v37/ioffe15.html.

John Jumper, Richard Evans, Alexander Pritzel, Tim Green, Michael Figurnov, Olaf Ronneberger, Kathryn Tunyasuvunakool, Russ Bates, Augustin Žídek, Anna Potapenko, et al. Highly accurate protein structure prediction with alphafold. *Nature*, 596(7873):583–589, 2021.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. In Peter L. Bartlett, Fernando C. N. Pereira, Christopher J. C. Burges, Léon Bottou, and Kilian Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25: 26th Annual Conference on Neural Information Processing Systems 2012. Proceedings of a meeting held December 3-6, 2012, Lake Tahoe, Nevada, United States*, pages 1106–1114, 2012. URL https://proceedings.neurips.cc/paper/2012/hash/c399862d3b9d6b76c8436e924a68c45b-Abstract.html.

AI @ Meta Llama Team. The llama 3 herd of models. jul 2024.

Ekdeep Singh Lubana, Robert P. Dick, and Hidenori Tanaka. Beyond batchnorm: Towards a unified understanding of normalization in deep learning. In Marc'Aurelio Ranzato, Alina Beygelzimer, Yann N. Dauphin, Percy Liang, and Jennifer Wortman Vaughan, editors, *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, pages 4778–4791, 2021. URL https://proceedings.neurips.cc/paper/2021/hash/2578eb9cdf020730f77793e8b58e165a-Abstract.html.

Kyle Luther. Why batch norm causes exploding gradients. Blog post, 2020. URL https://kyleluther.github.io/2020/02/18/batchnorm-exploding-gradients.html.

James Martens and Roger B. Grosse. Optimizing neural networks with kronecker-factored approximate curvature, 2015. URL http://proceedings.mlr.press/v37/martens15.html.

James Martens, Andy Ballard, Guillaume Desjardins, Grzegorz Swirszcz, Valentin Dalibard, Jascha Sohl-Dickstein, and Samuel S. Schoenholz. Rapid training of deep neural networks without skip connections or normalization layers using deep kernel shaping. *CoRR*, abs/2110.01765, 2021. URL https://arxiv.org/abs/2110.01765.

Oyebade K. Oyedotun, Kassem Al Ismaeil, and Djamila Aouada. Training very deep neural networks: Rethinking the role of skip connections. *Neurocomputing*, 441:105–117, 2021. doi: 10.1016/J.NEUCOM.2021.02.S004. URL https://doi.org/10.1016/j.neucom.2021.02.004.

Jeffrey Pennington, Samuel S. Schoenholz, and Surya Ganguli. Resurrecting the sigmoid in deep learning through dynamical isometry: theory and practice. In Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 4785–4795, 2017. URL https://proceedings.neurips.cc/paper/2017/hash/d9fc0cdb67638d50f411432d0d41d0ba-Abstract.html.

Shibani Santurkar, Dimitris Tsipras, Andrew Ilyas, and Aleksander Madry. How does batch normalization help optimization? In Samy Bengio, Hanna M. Wallach, Hugo Larochelle, Kristen Grauman, Nicolò Cesa-Bianchi, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada*, pages 2488–2498, 2018. URL https://proceedings.neurips.cc/paper/2018/hash/905056c1ac1dad141560467e0a99e1cf-Abstract.html.

Andrew M. Saxe, James L. McClelland, and Surya Ganguli. Exact solutions to the nonlinear dynamics of learning in deep linear neural networks. In Yoshua Bengio and Yann LeCun, editors, *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*, 2014a. URL http://arxiv.org/abs/1312.6120.

Andrew M. Saxe, James L. McClelland, and Surya Ganguli. Exact solutions to the nonlinear dynamics of learning in deep linear neural networks. In Yoshua Bengio and Yann LeCun, editors, *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*, 2014b. URL http://arxiv.org/abs/1312.6120.

Samuel S. Schoenholz, Justin Gilmer, Surya Ganguli, and Jascha Sohl-Dickstein. Deep information propagation. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017. URL https://openreview.net/forum?id=H1W1UN9gg.

Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In Yoshua Bengio and Yann LeCun, editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015. URL http://arxiv.org/abs/1409.1556.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 5998–6008, 2017. URL https://proceedings.neurips.cc/paper/2017/hash/3f5ee243547dee91fbd053c1c4a845aa-Abstract.html.

Andreas Veit, Michael J. Wilber, and Serge J. Belongie. Residual networks behave like ensembles of relatively shallow networks. In Daniel D. Lee, Masashi Sugiyama, Ulrike von Luxburg, Isabelle Guyon, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain*, pages 550–558, 2016. URL https://proceedings.neurips.cc/paper/2016/hash/37bc2f75bf1bcfe8450a1a41c200364c-Abstract.html.

Lechao Xiao, Yasaman Bahri, Jascha Sohl-Dickstein, Samuel S. Schoenholz, and Jeffrey Pennington. Dynamical isometry and a mean field theory of cnns: How to train 10, 000-layer vanilla convolutional neural networks. In Jennifer G. Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, volume 80 of *Proceedings of Machine Learning Research*, pages 5389–5398. PMLR, 2018. URL http://proceedings.mlr.press/v80/xiao18a.html.

Greg Yang and Samuel S. Schoenholz. Mean field residual networks: On the edge of chaos. In Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 7103–7114, 2017a. URL https://proceedings.neurips.cc/paper/2017/hash/81c650caac28cdefce4de5ddc18befa0-Abstract.html.

Greg Yang and Samuel S. Schoenholz. Mean field residual networks: On the edge of chaos. In Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 7103–7114, 2017b. URL https://proceedings.neurips.cc/paper/2017/hash/81c650caac28cdefce4de5ddc18befa0-Abstract.html.

Greg Yang, Jeffrey Pennington, Vinay Rao, Jascha Sohl-Dickstein, and Samuel S. Schoenholz. A mean field theory of batch normalization. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019a. URL https://openreview.net/forum?id=SyMDXnCcF7.

Greg Yang, Jeffrey Pennington, Vinay Rao, Jascha Sohl-Dickstein, and Samuel S. Schoenholz. A mean field theory of batch normalization. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019b. URL https://openreview.net/forum?id=SyMDXnCcF7.

Guodong Zhang, Aleksandar Botev, and James Martens. Deep learning without shortcuts: Shaping the kernel with tailored rectifiers. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net, 2022. URL https://openreview.net/forum?id=U0k7XNTiFEq.

Hongyi Zhang, Yann N. Dauphin, and Tengyu Ma. Fixup initialization: Residual learning without normalization. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019. URL https://openreview.net/forum?id=H1gsz30cKX.

A Theory Appendix

A.1 Possible Reparametrizations

In this Section, we show that we can find a non-equivalent reparametrization of a residual block as a feedforward block. The purpose is to show that while some reparametrizations exist, they come at the cost of additional assumptions and/or extending the network's width. This therefore does not contradict the main result in Section 1.

A.1.1 Arbitrary Nonlinearity

Proposition 2 (Locally Linear Nonlinearity)**.** *Let R be a residual block defined as follows:*

$$R(x) = \phi(\overline{W}x + \bar{b}) + x \tag{8}$$

where $\overline{W} \in \mathbb{R}^{n\times n}$ represents the weight matrix and $\bar{b} \in \mathbb{R}^n$ the bias and $\phi : \mathbb{R} \to \mathbb{R}$ is an element-wise nonlinear function, which is twice differentiable at some point $c \in \mathbb{R}$ and satisfies $\phi'(c) \neq 0$.

R can then be approximated uniformly for a bounded input domain by a feedforward layer F with one additional linear layer and double the width:

$$F(x) = W_2\phi(W_1x + b_1) + b_2, \tag{9}$$

with weights $W_1 \in \mathbb{R}^{2n\times n}$, $W_2 \in \mathbb{R}^{n\times 2n}$ and biases $b_1 \in \mathbb{R}^{2n}$, $b_2 \in \mathbb{R}^n$.

Proof. In a first step, we need to shift and shrink the input into the linear region of ϕ using an affine map $L_1 : \mathbb{R}^n \to \mathbb{R}^{2n}$. Let $\varepsilon > 0$; we can then write:

$$L_1(x) \;\coloneqq\; \underbrace{\begin{bmatrix} \varepsilon \cdot I \\ \overline{W} \end{bmatrix}}_{\mathbb{R}^{2n\times n}} x + \underbrace{\begin{bmatrix} c \cdot \mathbb{1} \\ \bar{b} \end{bmatrix}}_{\mathbb{R}^{2n}}. \tag{10}$$

Next, after applying the nonlinear layer ϕ, we need to add, un-shrink, and shift back using $L_2 : \mathbb{R}^{2n} \to \mathbb{R}^n$:

$$L_2(x) \;\coloneqq\; \underbrace{\big[\frac{1}{\varepsilon\,\phi'(c)} \cdot I \;\; I\big]}_{\mathbb{R}^{n\times 2n}} x \;-\; \underbrace{\frac{\phi(c)}{\varepsilon\phi'(c)} \cdot \mathbb{1}}_{\mathbb{R}^n}. \tag{11}$$

We can now put together our feedforward layer $F = L_2 \circ \phi \circ L_1$ and obtain:

$$F(x) = \frac{\phi(\varepsilon x + c)}{\varepsilon\phi'(c)} + \phi(\overline{W}x + \bar{b}) \;-\; \frac{\phi(c)}{\varepsilon\phi'(c)} \cdot \mathbb{1}. \tag{12}$$

If we now plug the Taylor expansion for the term we want to linearize

$$\phi(\varepsilon x + c) = \phi(c) \cdot \mathbb{1} + \phi'(c)\varepsilon x + \mathcal{O}(\varepsilon^2\|x\|^2) \tag{13}$$

into Equation 12, we finally obtain the reparametrization, which converges uniformly as $\varepsilon \to 0$:

$$F(x) = \frac{1}{\varepsilon\phi'(c)}\left(\phi(c)\mathbb{1} + \varepsilon\phi'(c)x + \mathcal{O}(\varepsilon^2\|x\|^2)\right) + \phi(\overline{W}x + \bar{b}) - \frac{\phi(c)}{\varepsilon\phi'(c)}\mathbb{1} \tag{14}$$

$$= x + \phi(\overline{W}x + \bar{b}) + \mathcal{O}(\varepsilon\|x\|^2). \tag{15}$$

□

In Appendix Section A.1.2, we show a similar construction specifically for $\phi = ReLU$ that works by shifting the data term into the linear region of the activation function instead of shrinking, thus not suffering from numerical issues related to the shrinking/unshrinking of the pre-activations.

A.1.2 RELU NONLINEARITY

Proposition 3 (ReLU Nonlinearity)**.** *Let R be a residual block defined as follows:*

$$R(x) = \phi(\overline{W}x + \bar{b}) + x \tag{16}$$

where $\overline{W} \in \mathbb{R}^{n\times n}$ represents the weight matrix and $\bar{b} \in \mathbb{R}^n$ the bias and $\phi : \mathbb{R} \to \mathbb{R}$ is the ReLU function.

We assume that all pre-activations are lower bounded by a constant, i.e. $\exists c \in \mathbb{R} : c < x_i \quad \forall i \in \{1, \dots, n\}$.

R can then be reparametrized by a feedforward layer F with one additional linear layer with double the width:

$$F(x) = W_2\phi(W_1x + \tilde{b}_1) + \tilde{b}_2, \tag{17}$$

with weights $W_1 \in \mathbb{R}^{2n\times n}$, $W_2 \in \mathbb{R}^{n\times 2n}$ and biases $\tilde{b}_1 \in \mathbb{R}^{2n}$, $\tilde{b}_2 \in \mathbb{R}^n$.

Proof. We construct our feedforward block F as follows:

$$F : x \xrightarrow{L_1} \begin{bmatrix} x + b_1 \\ \overline{W}x + \bar{b} \end{bmatrix} \xrightarrow{\phi} \begin{bmatrix} \phi(x + b_1) \\ \phi(\overline{W}x + \bar{b}) \end{bmatrix} \xrightarrow{L_2} \phi(x + b_1) + \phi(\overline{W}x + \bar{b}) + \tilde{b}_2, \tag{18}$$

where we decompose the full bias as

$$\tilde{b}_1 = \begin{bmatrix} b_1 \\ \bar{b} \end{bmatrix}. \tag{19}$$

Let $L_1 : \mathbb{R}^n \to \mathbb{R}^{2n}$ be an affine map:

$$L_1(x) \coloneqq \underbrace{\begin{bmatrix} I \\ \overline{W} \end{bmatrix}}_{\mathbb{R}^{2n\times n}} x + \underbrace{\begin{bmatrix} b_1 \\ \bar{b} \end{bmatrix}}_{\mathbb{R}^{2n}}, \tag{20}$$

and where $I \in \mathbb{R}^{n\times n}$ denotes the identity matrix. Let $L_2 : \mathbb{R}^{2n} \to \mathbb{R}^n$ be a second affine map:

$$L_2(x) \coloneqq \underbrace{\begin{bmatrix} I & I \end{bmatrix}}_{\mathbb{R}^{n\times 2n}} x + \underbrace{\tilde{b}_2}_{\mathbb{R}^n}. \tag{21}$$

We can now set the biases $b_1, \tilde{b}_2 \in \mathbb{R}^n$ with the following shifts:

$$b_1 \coloneqq -c \cdot \mathbb{1} \tag{22}$$

$$\tilde{b}_2 \coloneqq c \cdot \mathbb{1}, \tag{23}$$

where $\mathbb{1} \in \mathbb{R}^n$ denotes the all-ones vector. This way, we shift the pre-activations out of the nonlinear range of the activation function and back, and obtain an identity map:

$$\phi(x + b_1) + \tilde{b}_2 = x. \tag{24}$$

As we now obtain $F(x) = R(x)$, we have found a reparametrization. □

This construction avoids numerical issues due to shrinking/unshrinking of the activations present in Proposition 2, but requires a bias term to work (necessary for shifting the pre-activations) which is not present in a standard ResNet, for example.

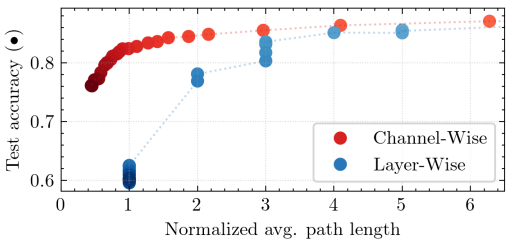


Figure 5: Comparing the test accuracies on **Cifar10** of partially linearized networks using a channel-wise or layer-wise partial linearization approach.

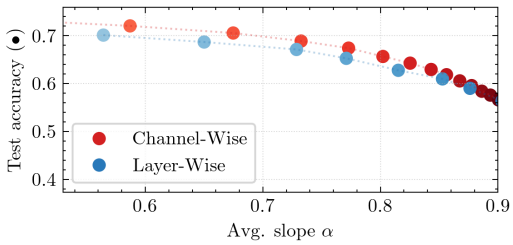


Figure 6: Comparing the test accuracies on **ImageNet** of partially linearized networks using a channel-wise or layer-wise partial linearization approach, using the **average slope** measure.

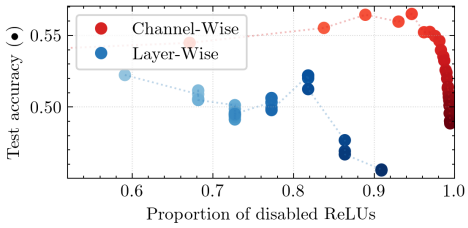


Figure 7: Comparing the test accuracies on **Cifar100** of partially linearized networks using a channel-wise or layer-wise partial linearization approach, using the **percentage of disabled units** measure.

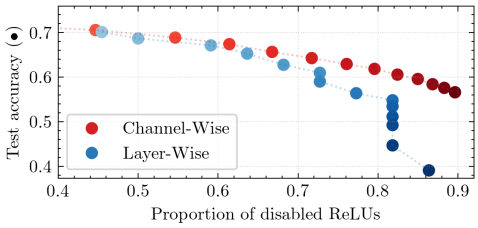


Figure 8: Comparing the test accuracies on **ImageNet** of partially linearized networks using a channel-wise or layer-wise partial linearization approach, using the **proportion of disabled units** measure.

A.1.3 Limitations

The constructions above however require significant assumptions, as they either yield an error term in Section A.1.1 or require the ReLU and bounded activations in Section A.1.2; both constructions require doubling the network's depth and width.

We don't see the halving of the number of layers as a major limitation, since we would typically compare networks with the same normalized average path length, i.e. a feedforward network with ℓ layers compared with a ResNet with approximately $\ell/2$ layers. Especially in normalized networks, the boundedness of the pre-activations does not seem like a major constraint. However, the effective halving of the width of the network is likely to affect the performance significantly.

In conclusion, it is impossible to reparametrize a residual network as an equivalent feedforward network, and possible reparametrizations we visited here require doubling the network's width.

B Experimental Appendix

In this Section, we present complementary experimental results to the main paper.

B.1 Partial Linearization on Cifar 10

We repeat the experiments from Section 4 on Cifar10 and show the results in Figure 5. We observe similar results to the ImageNet results discussed in the main paper; however, the NAPL, where the performance of the two techniques approximately breaks even, seems to be lower than for the ImageNet dataset: at around 4 NAPL. It appears that the harder the dataset, the higher the break-even point is.

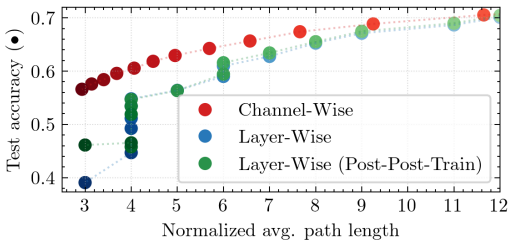


Figure 9: Comparing the test accuracies on **ImageNet** of partially linearized networks using a channel-wise or layer-wise partial linearization approach, including "post-post-training" runs, where we replace any non fully linear layer-wise PReLUs with unregularized channel-wise PReLUs and train for another 10 epochs to account for missing parameters.

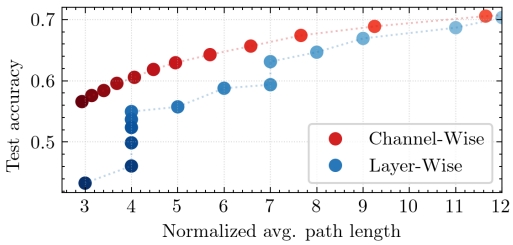


Figure 10: Comparing the test accuracies on **ImageNet** of partially linearized networks using a channel-wise or layer-wise partial linearization approach, with an additional **channel-wise multiplication layer** after each nonlinearity to account for missing parameters in the layer-wise case.

B.2 Partial Linearization with Different Measures

In this Section, we replicate the experiment from Figure 1, but we plot the average slope $\alpha = \frac{1}{n}\sum_{i=1}^{n}\alpha_i$ instead of the NAPL measure. We do this to confirm that the performance gap we observed cannot be attributed to a discretization artifact, as for the NAPL measure, a single channel is either fully linear or nonlinear. In Figure 6, we see that even using the average slope as a measure of nonlinearity, we can observe the same phenomenon, as the networks linearized using the layer-wise approach have consistently lower performances compared to the ones linearized using the channel-wise approach.

In Figure 8 and 7, we recreate the experiments of Figure 1 but show the results with the proportion of disabled PReLU units instead of the NAPL measure to confirm that the measured gap is not an artifact of this measure. As we can still observe a significant gap, this does not appear to be the case.

B.3 Accounting for Missing Parameters

In this Section, we address the concern that the results from Section 4 could be biased, since the networks extracted via the channel-wise method contain slightly more parameters (one slope parameter for each channel in a layer) than the networks extracted via the layer-wise method (one slope parameter per layer).

Sadly, this problem is not easily fixable via construction, as for example adding a linear layer before or after the nonlinearity would not be equivalent, but adding another non-regularized channel-wise PReLU unit before the layer-wise unit would add more nonlinear layers and thus void the experiments.

B.4 Post-Post-Training

Starting from the post-trained networks in the layer-wise case, we replace all layer-wise PReLU units that are not fully linear ($\alpha_i < 1$) by channel-wise units (with the existing slope as initialization parameter) and train again for 10 epochs, as in the original post-training, but without the regularization term. In Figure 9, we see that this yields just very slightly higher values and does not significantly reduce the effect strength.

B.5 Channel-Wise Multiplier

We add a channel-wise multiplication layer after the layer-wise PReLU layer to account for missing parameters and repeat the runs for the layer-wise linearization. Comparing Figure 10 to Figure 1(left), we do not observe a significant reduction in the performance gap between the two types of extracted sub-networks.

Training	CIFAR-10 / CIFAR-100
Epochs	200
Scheduler	Multistep ($\gamma = 0.1$)
Milestones	100, 150
Learning rate	0.1
Batch size	256
Optimizer	SGD + Momentum
Momentum	0.9
Weight decay	0.0001
Augmentation	Random Flip

Figure 11: Details of the training regime.

Post-Training	CIFAR-10 / CIFAR-100
Epochs	60
Scheduler	Multistep ($\gamma = 0.1$)
Milestones	40, 50
Learning rate	0.01
Batch size	256
Optimizer	SGD + Momentum
Momentum	0.9
Weight decay	0.0001
Augmentation	Random Flip

Post-Training	ImageNet
Epochs	10
Scheduler	Multistep ($\gamma = 0.1$)
Milestones	6, 8
Learning rate	0.01
Batch size	256
Optimizer	SGD + Momentum
Momentum	0.9
Weight decay	0.0001
Augmentation	Center Crop (224 px.)

Figure 12: Details of the post-training regime.

C Hardware and Training Details

The experiments in the paper were run on computers running Arch Linux, Python 3.13.3, PyTorch Version 2.7.0. The GPUs used were NVIDIA GeForce GTX 1080 Ti, GeForce RTX 2080 Ti and GeForce RTX 4090. The ImageNet post-training runs from Section 4 take approximately two days on an NVIDIA GeForce RTX 4090.

The hyper-parameters in Figure 11 were used to (post-)train on CIFAR-10 and CIFAR-100, reaching a base test-accuracy of approximately 87.5% on CIFAR-10 and 54.6% on Cifar100. As for the ImageNet runs, we used a pre-trained model from the RepVGG repository. For post-training, the hyper-parameters in Figure 12 were used.