Make Deep Networks Shallow Again

Bernhard Bermeitinger, Tomas Hrycej, and Siegfried Handschuh

Institute of Computer Science, University of St.Gallen (HSG),
St.Gallen, Switzerland

15 Sep 2023

Abstract

Deep neural networks have a good success record and are thus viewed as the best architecture choice for complex applications. Their main shortcoming has been, for a long time, the vanishing gradient which prevented the numerical optimization algorithms from acceptable convergence. An important special case of network architecture, frequently used in computer vision applications, consists of using a stack of layers of the same dimension. For this architecture, a breakthrough has been achieved by the concept of *residual connections*—an identity mapping parallel to a conventional layer. This concept substantially alleviates the vanishing gradient problem and is thus widely used. The focus of this paper is to show the possibility of substituting the deep stack of residual layers with a shallow architecture with comparable expressive power and similarly good convergence properties. A stack of residual layers can be expressed as an expansion of terms similar to the Taylor expansion. This expansion suggests the possibility of truncating the higher-order terms and receiving an architecture consisting of a single broad layer composed of all initially stacked layers in parallel. In other words, a sequential deep architecture is substituted by a parallel shallow one. Prompted by this theory, we investigated the performance capabilities of the parallel architecture in comparison to the sequential one. The computer vision datasets MNIST and CIFAR10 were used to train both architectures for a total of 6,912 combinations of varying numbers of convolutional layers, numbers of filters, kernel sizes, and other meta parameters. Our findings demonstrate a surprising equivalence between the deep (sequential) and shallow (parallel) architectures. Both layouts produced similar results in terms of training and validation set loss. This discovery implies that a wide, shallow architecture can potentially replace a deep network without sacrificing performance. Such substitution has the potential to simplify network architectures, improve optimization efficiency, and accelerate the training process.

1 Introduction

Deep neural networks (i.e., networks with many nonlinear layers) are widely considered to be the most appropriate architecture for mapping complex dependencies such as those

1

arising in Artificial Intelligence tasks. Their potential to map intricate dependencies has advanced their widespread use.

For example, the study (Meir et al., 2023) compares the first deep convolutional network for image classification with two sequential convolutional layers *LeNet* (LeCun et al., 1989) to its deeper evolution *VGG16* (Simonyan and Zisserman, 2015) with 13 sequential convolutional layers. While the performance gain in this comparison was significant, further increasing the depth resulted in very small performance gains. Adding three additional convolutional layers to *VGG16* improved the validation error slightly from 25.6 % to 25.5 % on the ILSVRC-2014 competition dataset (Russakovsky et al., 2015) while increasing the number of trainable parameters from 138M to 144M.

However, training these networks remains a significant challenge, often navigated through numerical optimization methods based on the gradient of the loss function. In deeper networks, the gradient can significantly diminish particularly for parameters distant from the output, leading to the well-documented issue known as the "vanishing gradient".

A breakthrough in this challenge is the concept of *residual connections*: using an identity function parallel to a layer (He et al., 2016). Each residual layer consists of an identity mapping copying the layer's input to its output and a conventional weighted layer with a nonlinear activation function. This weighted layer represents the residue after applying the identity. The output of the identity and the weighted layer are summed together, forming the output of the residual layer. The identity function plays the role of a bridge—or "highway" (Srivastava et al., 2015)—transferring the gradient w.r.t. layer output into that of the input with unmodified size. In this way, it increases the gradient of layers remote from the output.

The possibility of effectively training deep networks led to the widespread use of such residual networks and to the belief that this is the most appropriate architecture type (Mhaskar et al., 2017). However, extremely deep networks such as *ResNet-1000* with ten times more layers than *ResNet-101* (He et al., 2016) often demonstrate a performance decline.

Although there have been suggestions for wide architectures like *EfficientNet* (Tan and Le, 2019), these are still considered "deep" within the scope of this paper.

This paper questions the assumption that deep networks are inherently superior, particularly considering the persistent gradient problems. Success with methods like residual connections can be mistakenly perceived as validation of the superiority of deep networks, possibly hindering exploration into potentially equivalent or even better-performing "shallow" architectures.

To avoid such premature conclusions, we examine in this paper the relative performance of deep networks over shallow ones, focusing on a parallel or "shallow" architecture instead of a sequential or "deep" one. The basis of the investigation is the mathematical decomposition of the mapping materialized by a stack of convolutional residual networks into a structure that suggests the possibility of being approximated by a shallow architecture. By exploring this possibility, we aim to stimulate further research, opening new avenues for AI architecture exploration and performance improvement.

2 Decomposition of stacked residual connections

A layer of a conventional multilayer perceptron can be thought of as a mapping $y = F_h(x)$. With the residual connection concept (He et al.,

2016), this mapping is modified to

$$y = Ix + F_h(x) \tag{1}$$

For the h-th hidden layer, the recursive relationship is

$$z_h = Iz_{h-1} + F_h(z_{h-1}) \tag{2}$$

For example, the second and the third layers can be expanded to

$$z_2 = Iz_1 + F_2(z_1) \tag{3}$$

and

$$\begin{aligned} z_3 &= Iz_2 + F_3(z_2) \\ &= I(Iz_1 + F_2(z_1)) + F_3(Iz_1 + F_2(z_1)) \\ &= Iz_1 + F_2(z_1) + F_3(Iz_1 + F_2(z_1)) \end{aligned} \tag{4}$$

In the operator notation, it is

$$z_h = z_{h-1} + F_h * z_{h-1} = (I + F_h) * z_{h-1} \tag{5}$$

For linear operators, the recursion up to the final output vector y can be explicitly expanded (Hrycej et al., 2023, Section 6.7.3.1)

$$y = I * x + \sum_{h=1}^{H} F_h * x + \sum_{h=1}^{H} \sum_{k=1, k>h}^{H} F_k * F_h * x \cdots \tag{6}$$

with all combinations of operator triples, quadruples, etc. up to the product of all H layer operators.

Typically, these layer mappings are not linear due to their activation functions such as *sigmoid*, *tanh*, or *ReLU*. As a result, it does not satisfy the condition $F_h(x+z) = F_h(x) + F_h(z)$. However, their gradient is a linear operator. In a multilayer perceptron with a residual connection, the error gradient w.r.t. the output of the h-th layer is

$$\begin{aligned} \frac{\partial E}{\partial z_h} &= \left(\prod_{k=h+1}^{H} \frac{\partial z_k}{\partial z_{k-1}} \right) \frac{\partial E}{\partial z_H} \\ &= \left(\prod_{k=h+1}^{H} \left(I + W_k^T \nabla F_k \right) \right) \frac{\partial E}{\partial z_H} \end{aligned} \tag{7}$$

The error gradient w.r.t. the weights is, for both standard layers and those with residual connection

$$\frac{\partial E}{\partial W_h} = \nabla F_h \frac{\partial E}{\partial z_h} z_{h-1}^T \tag{8}$$

and w.r.t. biases

$$\frac{\partial E}{\partial b_h} = \nabla F_h \frac{\partial E}{\partial z_h} \tag{9}$$

This shows that the expansion given in Eq. (6) is valid for an approximation linearized with the help of the local gradient. In particular, it is valid around the minimum.

In an analogy to Taylor expansion, it can be hypothesized that the first two terms

$$y = I * x + \sum_{h=1}^{H} F_h * x \tag{10}$$

may be a reasonable approximation of the whole mapping in Eq. (6).

In terms of implementation as neural networks, the stack of layers with residual connections (as exemplified in Fig. 1) could be approximated by the parallel architecture such as that illustrated in Fig. 2.

Of course, this hypothesis has to be confirmed by tests on real-world problems. If acceptable, it would be possible to substitute a deep residual network of H sequential layers with a "shallow" network with a single layer consisting of H individual modules in parallel, summing their output vectors. Each of these modules would be equivalent to one layer in the deep architecture. The main objective is not to prove that both networks are nearly equivalent with the same parameter set, as this is unlikely to be the case. Rather, the goal is to demonstrate that both shallow and deep architectures can effectively learn and attain comparable performances on the given task. The consequence would be that the shallow architecture can perform as well as the deep one, with the same number of parameters. This may be
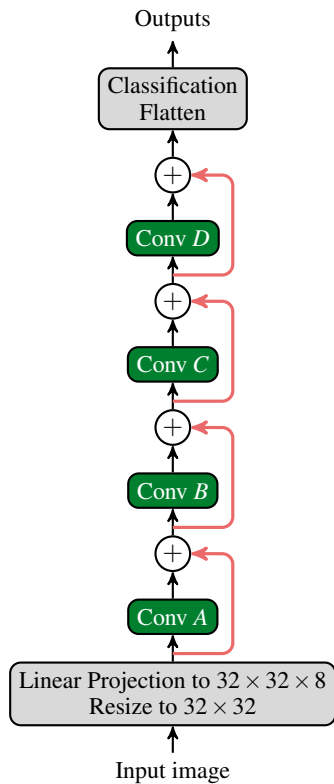
3

Figure 1: Overview of the sequential architecture with four consecutive convolutional layers with eight filters each and their skip connections.
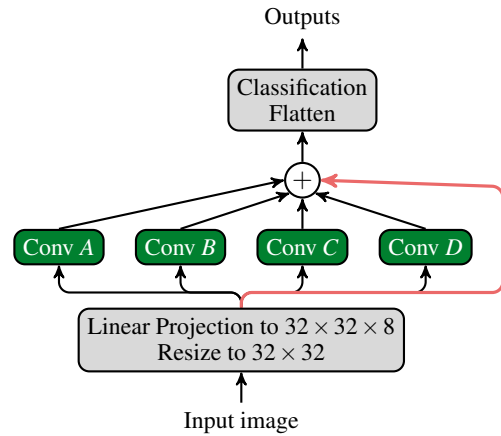

Figure 2: Overview of the parallelized architecture of Fig. 1 with four convolutional layers with eight filters each and one skip connection.

relevant for the preferences in setting up neural networks for particular tasks since shallow networks suffer less from numerical computing problems such as vanishing gradient.

3 Setup of computing experiments

The analysis of Section 2 suggests that the expressive power of a network architecture in which stacked residual connection layers of a deep network are reorganized into a parallel operation in a single, broad layer, may be close to that of the original deep network. This hypothesis is to be tested on practically relevant examples.

It is important to point out that residual connection layers are restricted to partial stacks of equally sized layers (otherwise the unity mapping could not be implemented). A typical use of such networks is image classification where an image is processed by consecutive layers of size equal to the (possibly reduced) pixel matrix. The output of this network is usually a

vector of class probabilities that differ in dimensionality from that of the input image. This is the reason for one or more non-residual layers at the output and some preprocessing non-residual layers at the input.

Residual connections can be used for any stack of layers of the same dimensions. However, the layers in domains such as image processing are mostly of the *convolutional* type. This is a layer concept in which the same, relatively small weight matrix, is applied to the neighbor environment of every position in the input. They are implementing a local operator (such as edge detection) shifted over the extension of the image. The following benchmark applications use convolutional layers.

Filters are a concept in convolutional layers that consist of a multiplicity of such convolution operators. Each filter convolves individually with the input matrix for generating the output. Multiple filters in a layer operate independently from each other, building a parallel structure. The computing experiments reported here were done both with and without multiple filters. The possibility of making the consecutive layer stack parallel concerns only the middle part with residual connections of identically sized layers.

For the experiments, the two well-known image classification datasets MNIST (LeCun et al., 1998) and CIFAR10 (Krizhevsky, 2009) were used. MNIST contains black and white images of handwritten digits (0–9) while CIFAR10 contains color images of exclusively ten different mundane objects like "horse", "ship", or "dog". They contain 60,000 (MNIST) and 50,000 (CIFAR10) training examples. Their respective preconfigured test split of each 10,000 examples are used as validation sets. While CIFAR10 is evenly distributed among all classes, MNIST is roughly evenly distributed with a standard deviation of 322 for the training set and 59 for the validation set. We took no special treatment for this slight class imbalance.

A series of computing experiments of all the following possible architectures were run:

- Number of convolutional layers: 1, 2, 4, 8, 16, 32

- Number of filters per convolutional layer: 1, 2, 4, 8, 16, 32

- Kernel size of a filter: 1×1, 2×2, 4×4, 6×6, 8×8, 16×16

- Activation function of each convolutional layer: *sigmoid*, *ReLU*

Figure 1 shows the sequential architecture with depth 4 and 8 filters per convolutional layer. For comparison, the parallelized version is shown in Fig. 2. The sizes of the filters' kernels are not shown because they don't interfere with the layout.

The images are resized to 32×32 pixels to match the varying kernel sizes. For the summation of the skip connection and the convolutional layer to work out, they need to have the same dimensionality. Therefore, for preprocessing, the images are linearly mapped to match the convolutional layers' output dimensions. To keep the architecture simple and reduce the possibility of additional side effects, the input is flattened into a one-dimensional vector before the dense classification layer with ten linear output units. These linear layers are initialized with the same set of fixed random values throughout all experiments.

The same configuration setup was used for the number of parallel filters per layer. Parallel filters are popular means of extending a straightforward convolution layer architecture: instead of each layer being a single convolution of the previous layer, it consists of multiple convolution filters in parallel. In all well-performing image classifiers based on convolutional layers, multiple filters are

used (Fukushima, 1980; Krizhevsky et al., 2012; Simonyan and Zisserman, 2015).

Throughout all experiments, the parameters of the layers at the same depths were always initialized with the same random values with a fixed seed. For example, the two layers labeled *A* in Figs. 1 and 2 started their training from the same parameter set.

The categorical cross-entropy loss was employed as the loss function due to its suitability for multi-class classification problems. This loss served also as the main assessment of the training performance. While metrics like classification accuracy are more intuitive from an application point of view, it's important to align the assessment with the loss function being optimized. The convergence of the optimization process can only be measured with the help of the minimized loss function. Although other metrics are valuable in the context of the application, they may not exhibit a direct monotonic relationship with the loss function, making them less suitable for comparing learning convergence, especially when both compared architectures share identical parameter sets. Thus, our choice of using cross-entropy loss as the performance metric is justified.

The datasets were not shuffled between epochs or experiments, leading to identical batches throughout all experiment runs.

As the optimizer, RMSprop (Hinton, 2012) was chosen with a fixed learning rate throughout all steps with a batch size of 512. All experiments were duplicated for the learning rates 10^{-2}, 10^{-3}, 10^{-4}, and 10^{-5}. Different learning rates had only a marginal effect on the results, the we only report the results obtained with a learning rate of 10^{-4}.

Each experiment ran for 100 epochs, which resulted in 11,800 optimization steps for MNIST, and 9,800 steps for CIFAR10. The 6,912 experiments were run individually on *NVIDIA Tesla V100* GPUs for a total run time of 79 days. The results are reported for the kernel size 16×16 which showed the best average classification performance although not significantly different.

4 Computing experiments

4.1 With a single filter

The losses after the 100 epochs for the training set (T) and the validation set (V) are given in Fig. 3. The performance of both architectures can be observed by the points on the green (sequential architecture) and red (parallel variant) points. The solid lines represent the training loss and the dashed lines the validation loss.

Due to their identical layout and equal random initialization, training the two networks with one convolutional layer and one filter each resulted consequently in equal loss values.

It can be observed that both architectures perform similarly, in particular for the largest depths of 16 and 32. For MNIST, the shallow, parallel architecture slightly outperforms the original, sequential one, while the relationship is inverse for the CIFAR10 dataset.

4.2 With multiple filters

A single-filter architecture is the most transparent one but it is scarcely used. It is mostly assumed that more filters are necessary to reach the desired classification performance. Therefore, experiments with multiple (1 to 32) filters per convolutional layer are included.

Same as before, the results after training for 100 epochs are shown in Figs. 4a and 4b. They show an interesting development for CIFAR10: the training loss decreases by raising the number of filters while the validation loss largely increases for more than four filters. The validation loss considerably deteriorates for the sequential architecture. (The re-
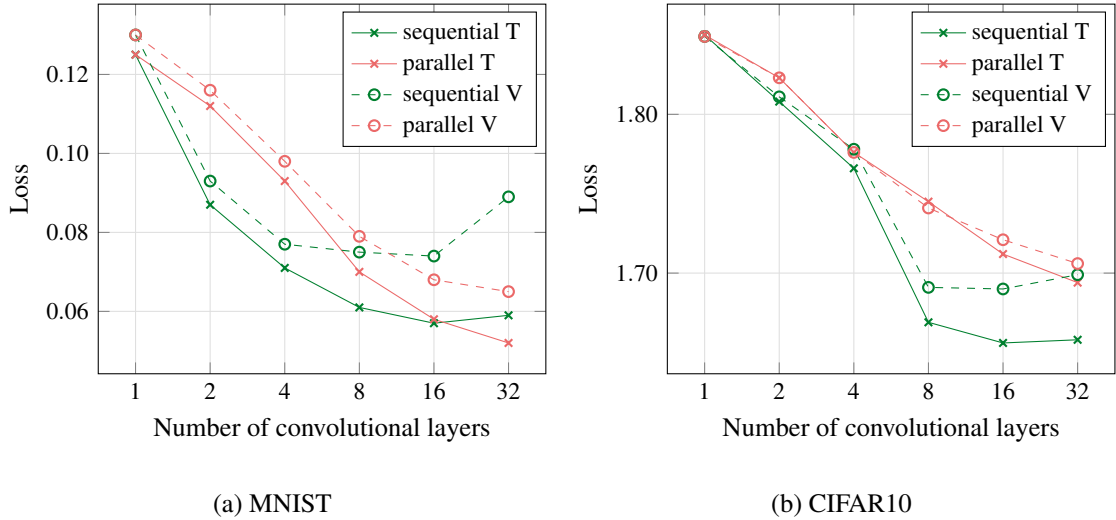
(a) MNIST

(b) CIFAR10

Figure 3: Sequential vs. parallel architecture: loss dependence on the number of residual convolutional layers (with a single filter per layer) for the two datasets MNIST (left) and CIFAR10 (right)

sults for MNIST are similar for the training set but less interpretable for the validation set.)

The reason for the distinct picture on CIFAR10 is to be sought in relationships between constraints imposed by the task and the number of free trainable parameters (Hrycej et al., 2023, Chapter 4). A task with $K = 50{,}000$ training examples constitutes equally many constraints (resulting from the goal to accurately match the target values) for each output value. For 10 classes, there are $M = 10$ such output values whose reference values are to be correctly predicted by the classifier. This creates KM constraints (here: $50{,}000 \times 10 = 500{,}000$). For the mapping represented by the network, there are P free (i.e., mutually independent) parameters to make the mapping satisfy the constraints.

- With $P = KM$, the system is perfectly determined and could be solved exactly.

- With $P > KM$, the system is underdetermined. A part of the parameters is set to arbitrary values so that novel examples

from the validation set receive arbitrary predictions.

- With $P < KM$, the system is overdetermined, and not all constraints can be satisfied. This may be useful if the data are noisy, as it is not desirable to fit to noise.

An appropriate characteristic is the overdetermination ratio Q from (Hrycej et al., 2022) defined as

$$Q = \frac{KM}{P} \tag{11}$$

The number of genuinely free parameters is difficult to figure out. It can only be approximated by the total number of parameters, keeping in mind that the number of actually free parameters can be lower.

In training a model by fitting to data, the presence of the noise has to be considered. The model should reflect the underlying genuine laws in the data but not the noise. Fitting to the latter is undesirable and is the substance of the well-known phenomenon of *overfitting*. It was shown in (Hrycej et al., 2023, Chapter 4)

7
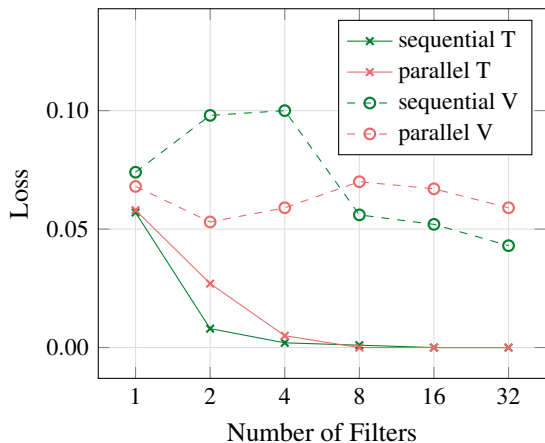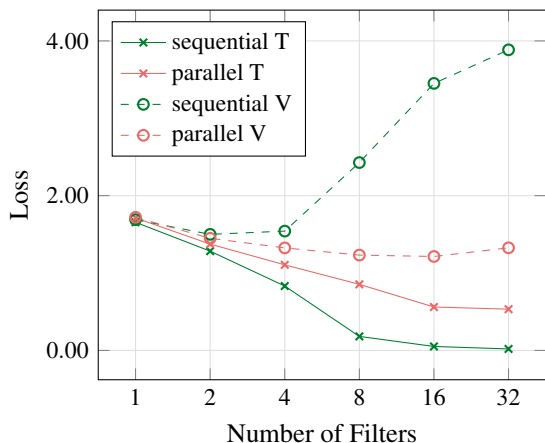
(a) MNIST

(b) CIFAR10

Figure 4: Sequential vs. parallel architecture: loss dependence on the number of filters (with 16 convolutional layers) for the two datasets MNIST (top) and CIFAR10 (bottom)

Table 1: Overdetermination ratios for both datasets and different model sizes based on the number of filters for each of the 16 convolutional layers

		Overdetermination ratio Q	
#filters	#parameters	MNIST	CIFAR10
1	14k	41.771	34.804
2	37k	16.256	13.545
4	106k	5.630	4.691
8	344k	1.743	1.453
16	1.2M	0.495	0.412
32	4.5M	0.132	0.110

that fitting to the additive noise and thus the influence of training set noise to the model prediction is reduced to the fraction $1/Q$. In other words, it is useful to keep the overdetermination ratio Q significantly over 1.

This supplementary information for the plotted variants is given in Table 1. Acceptable values of the overdetermination ratio Q are given with filter counts of 1, 2, and 4. This is consistent with the finding that overfitting did not take place in single-filter architectures presented in Section 4.1.

For 8 filters or more, Q is close to 1 or even below it. In this group, the validation loss can grow arbitrarily although the training loss is reduced. This is the result of arbitrarily assigned values of underdetermined parameters.

Altogether, the parallel architecture shows better performance on the validation set despite the slightly inferior loss on the training set. This can be attributed rather to the random effects of underdetermined parameters than to the superiority of one or other architecture. In this sense, both architectures can be viewed as approximately equivalent concerning their representational capacity.

4.3 Trade-off of the number of filters and the number of layers

As an additional view to the relationship between the depth and the width of the network, a group of experiments is analyzed in which the product of the number of filters (F) and the number of convolution layers (C) are kept constant. In this way, also "intermediary" architectures between deep and shallow ones are captured. For example, an architecture with 32 filters and a single convolutional layer has a ratio of $1/32$ while the ratio with one filter and 32 layers is $32/1$. For 16 layers with each 8 filters, it is $16/8 = 2$.

For the product of 32, there are the following combinations of $C \times F$: 1×32, 2×16, 4×8, 8×4, 16×2 and 32×1. In Fig. 5, they are ordered along their depth-width ratio C/F: $1/32$, $2/16$, $4/8$, $8/4$, $16/2$, and $32/1$. These architectures are represented by the green curves.

As a reference, the red curve shows their shallow counterparts. Those are all single-layer architectures. They differ only in the number of parameters, consistent with their sequential counterparts represented by the green curve. The difference in the number of parameters is due to the different sizes of the classification layer following the residual connection sequence. This classification layer is broader for more filters as its input is larger the more filters there are.

Both the training and validation losses increase with the depth-width ratio, indicating the superiority of the shallow architectures. However, it is important to note that this comparison may not be completely fair due to the inherent difference in parameter numbers. Specifically, variants with higher depth-width ratios have a diminishing number of parameters resulting from their smaller number of filters.

In Figs. 5a and 5b, it can be observed that the training loss for flattened alternatives is slightly larger compared to the other architectures. However, the validation loss for flattened alternatives is smaller, albeit to a moderate extent.

In summary, the deep variants can certainly not be viewed as superior in overall terms. Both architectures are roughly equivalent, as long as the number of parameters is equal.

5 Statistics of experiments

In addition to experiment runs selected for the presentation in the previous sections, statistics over all 6,912 runs, partitioned into some categories, may be useful to complete the performance picture. Of course, averaging hundreds to thousands of experiments does not guarantee to reflect all theoretical expectations succinctly; it can only confirm rough trends.

This statistical summary is presented in Table 2. The losses for training and validation as well as for sequential and parallel architectures are partitioned into intervals of overdetermination ratio to show the different behavior.

According to the theory, with a growing overdetermination ratio, the discrepancy between training and validation loss becomes smaller. On the other hand, larger overdetermination ratios imply smaller numbers of free network parameters. Sometimes, this leads to increased losses from the diminished representation capacity of the network. For ratios smaller than 1, the validation loss may arbitrarily grow because of underdetermined parameters fitted to training data noise (*overfitting*). This arbitrary growth may be more or less articulated, depending mostly on random factors. However, there is always a considerable risk of such poor generalization.

As observed in the individual experiments presented, small discrepancies between training and validation loss are reached for overdetermination ratios larger than 3 for CIFAR10

Table 2: Mean training and validation loss for sequential and parallel architectures and various determination ratios Q intervals

| | $Q \in [0,1)$ | | $Q \in [1,3)$ | | $Q \in [3,10)$ | | $Q \in [10,\infty)$ | |
	train	val	train	val	train	val	train	val
MNIST								
sequential	0.00013	0.05201	0.01702	0.12449	0.03620	0.11743	0.11246	0.13550
parallel	0.00009	0.07551	0.02679	0.11468	0.05238	0.11467	0.13310	0.14900
CIFAR10								
sequential	0.25326	2.03107	0.72510	1.31691	1.07333	1.34721	1.58608	1.65354
parallel	0.52658	1.32386	0.88701	1.24884	1.17085	1.34227	1.63449	1.68879

and larger than 10 for MNIST. These small discrepancies testify to good generalization capability, expected for large overdetermination ratios.

With $Q < 1$, the validation loss deteriorates for CIFAR10 data if compared with the Q of the higher interval. This is the effect of arbitrary parameter values caused by underdetermination.

To summarize, there is a slight advance of shallow architectures for the validation set (five out of eight categories), and deep architectures are better on the training set. The training and validation losses are mostly closer together for the parallel architecture.

6 Conclusion

It is stated in Section 2 that a deep residual connection network can be approximately expanded into a sum of shorter (i.e., less deep) sequences of different orders. Truncating the expansion to the first two terms results in a shallow architecture with a single layer. This suggests a hypothesis that the representational capacity of such a shallow architecture may be roughly as large as that of the original deep architecture. If validated, this hypothesis could open avenues to bypass issues typically associated with deep architectures.

Subsequent computational experiments conducted on two widely recognized image classification tasks, MNIST and CIFAR10, seem to confirm this theoretically founded expectation. The performance of both architectures (in configurations with identical numbers of network parameters) is close to each other, with a slight advance of shallow architectures in terms of loss on the validation set.

While the deep architecture performed marginally better on the training set, the cause of its underperformance on the validation set remains an open question. It is plausible that the deep architecture's ability to capture abrupt nonlinearities may also make it prone to overfitting to noise. In contrast, the shallow network, due to its inherent smoothness, might exhibit a higher tolerance towards training set noise.

In conclusion, our results suggest a potential parity in the performance of deep and shallow architectures. It is important to note that the optimization algorithm utilized in this study is a first-order one, which lacks guaranteed convergence properties. Future research could explore the application of more robust second-order algorithms, which, while not commonly implemented in prevalent software packages, could yield more pronounced results. This work serves as a preliminary step towards
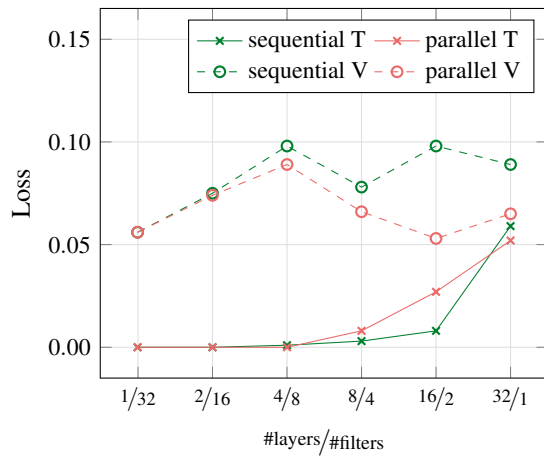
(a) MNIST
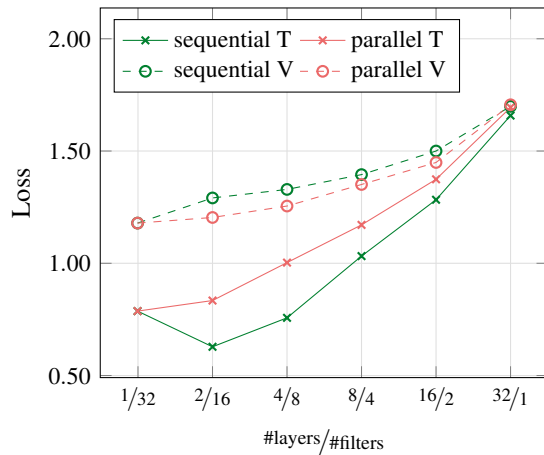
(b) CIFAR10

Figure 5: Sequential vs. parallel architecture: loss dependence on the ratio of the numbers of layers and filters (product of the number of layers and the number of filters is fixed at 32) for the two datasets MNIST (top) and CIFAR10 (bottom)

reevaluating architectural decisions in the field of neural networks, urging further exploration into the comparative efficacy of shallow and deep architectures.

References

Fukushima, K. (1980). Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. *Biological Cybernetics*, 36(4):193–202.

He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep Residual Learning for Image Recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, Las Vegas, NV, USA. IEEE.

Hinton, G. (2012). Neural Networks for Machine Learning.

Hrycej, T., Bermeitinger, B., Cetto, M., and Handschuh, S. (2023). *Mathematical Foundations of Data Science*. Texts in Computer Science. Springer International Publishing, Cham.

Hrycej, T., Bermeitinger, B., and Handschuh, S. (2022). Number of Attention Heads vs. Number of Transformer-encoders in Computer Vision. In *Proceedings of the 14th International Joint Conference on Knowledge Discovery, Knowledge Engineering and Knowledge Management*, pages 315–321, Valletta, Malta. SCITEPRESS.

Krizhevsky, A. (2009). Learning Multiple Layers of Features from Tiny Images. Dataset, University of Toronto.

Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). ImageNet Classification with Deep Convolutional Neural Networks. In *Advances in Neural Information Processing Systems*, volume 25. Curran Associates, Inc.

LeCun, Y., Boser, B., Denker, J. S., Henderson, D., Howard, R. E., Hubbard, W., and Jackel, L. D. (1989). Backpropagation Applied to Handwritten Zip Code Recognition. *Neural Computation*, 1(4):541–551.

LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324.

Meir, Y., Tevet, O., Tzach, Y., Hodassman, S., Gross, R. D., and Kanter, I. (2023). Efficient shallow learning as an alternative to deep learning. *Scientific Reports*, 13(1):5423.

Mhaskar, H., Liao, Q., and Poggio, T. (2017). When and Why Are Deep Networks Better Than Shallow Ones? *Proceedings of the AAAI Conference on Artificial Intelligence*, 31(1).

Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., Berg, A. C., and Fei-Fei, L. (2015). ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision*, 115(3):211–252.

Simonyan, K. and Zisserman, A. (2015). Very Deep Convolutional Networks for Large-Scale Image Recognition.

Srivastava, R. K., Greff, K., and Schmidhuber, J. (2015). Training very deep networks. In *Advances in Neural Information Processing Systems*. Curran Associates, Inc.

Tan, M. and Le, Q. (2019). EfficientNet: Rethinking model scaling for convolutional neural networks. In Chaudhuri, K. and Salakhutdinov, R., editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 6105–6114. PMLR.