CNNs can efficiently trace paths, too

Anonymous Author(s)
Affiliation
Address
email

Abstract

The role of feedback (or recurrent) connections is a fundamental question in neuroscience and machine learning. Recently, two benchmarks [1, 2], which require following paths in images, have been proposed as examples where recurrence was considered to be essential. In this work, we demonstrate that these tasks can equally well or even better be solved using efficient convolutional feed-forward neural networks (CNNs). We analyse ResNet training regarding model complexity and sample efficiency and show that a narrow, parameter-efficient ResNet performs on par with the computationally more complex hCNN and td+hCNN models from previous work.
Source code: `https://to-be-published`

1 Introduction

The proper use of recurrent processing could be crucial to enable neural networks to solve inherently iterative problems such as understanding subway maps or searching for complex patterns. While evidence from neuroscience suggests that human perception heavily relies on feedback connections, most successful computer vision models follow the feed-forward paradigm.

In an effort to determine the limits where recurrence is necessary, Linsley et al. [1] introduced the pathfinder challenge. Subsequently, the cABC dataset following a similar paradigm was proposed by Kim et al. [2]. In both tasks, dotted paths must be traced in order to decide if two points are connected via a path or pertain to different paths. Each image contains two points along with multiple distractions, and a learner must make a single binary decision for each image. Building upon earlier work using variants of gated recurrent units [3, 1], Kim et al. [2] present a convolutional recurrent neural network (hCNN) featuring horizontal connections, and its extension td+hCNN, which features horizontal connections and top-down feedback. They report that recurrent architectures perform well on pathfinder and cABC, respectively. In contrast, they observe that commonly used feed-forward architectures including ResNets and U-Nets fail at the highest difficulty level despite requiring magnitudes more trainable parameters than the proposed RNNs. Based on this result, the authors diagnose "a computational deficiency of feedforward networks."

Contributions In this work we demonstrate that, contrary to the claims of [1] and [2], recurrence is not necessary for these path-finding tasks and convolutional neural networks are capable of solving even the most challenging versions. We investigate the factors that enable ResNets to learn solving these tasks reliably. We found that the network width accounts for only a small fraction of performance and ResNets can be stripped off substantially, undercutting td+hCNN in terms of parameters and inference speed (the latter holds for all ResNets).

Related Work The question of when recurrence is required and how it is implemented is old and central to the fields of neuroscience and machine learning. For a long time, the primary choice for text processing were recurrent neural networks such as LSTM [4] or GRU [5], due to their ability to adapt

Submitted to 2nd Workshop on Shared Visual Representations in Human and Machine Intelligence (SVRHM).

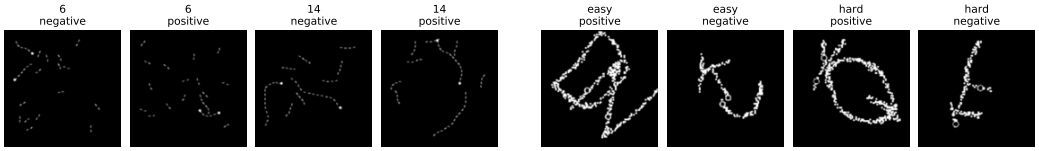

Figure 1: Four samples of different difficulty levels from the Pathfinder (left) and cABC datasets (right) each. Positive means a connections exists, while in negatives samples the dots are not connected.

the number of processing steps. However, it was found that temporal convolutional neural networks were able to solve these tasks [6]. Later, attention-based feed-forward architectures revolutionized the field [7]. For robotic path planning, CNNs were shown to work well, too [8].

2 Experiments

2.1 Pathfinder and cABC Datatset

The task of the *Pathfinder* dataset is to determine if two dots are connected by a curve (or path) or not (Fig. 1, left). Curves are dashed (i.e. composed of small segments) and can intersect. This task is designed to be solvable with Gestalt-based approaches. Both paths and dots are rendered into a single image. Hence the decision if a connection exists is a binary classification problem which we can report an accuracy for. Pathfinder consists of three datasets of varying difficulty, which is controlled by the path length: 6, 9 and 14 elements.

In follow-up work, Kim et al. [2] introduced the *Cluttered ABC (cABC)* dataset (Fig. 1, right), which shares the binary image classification format and the division into three levels of difficulty (easy, intermediate and hard). In contrast to Pathfinder, it has global semantics in form of letters, but is locally degenerate. The latter is due to paths being randomly sprinkled dots which follow a curve as an ensemble on a higher level. In order to follow paths here, dots must be grouped in a top-down manner. Indeed, [2] report that top-down feedback enables solving this task while they failed to train wide CNNs on this task.

2.2 Models

We focus our analysis on the ResNet model family, which has emerged as one of the predominant feed-forward CNN architectures. They use residual connections, which bypass computational blocks preventing vanishing gradients. Contrary to feedback connections, which induce information from higher levels of processing, residual connections are strictly feed-forward shortcuts. The original ResNet18 [9] designed for ImageNet [10] has around 11.2 million parameters.

2.2.1 Narrow ResNet18 (nRN18)

We adopt the ResNet18 architecture [9] but reduce the number of channels, since the synthetic images used in this paper are less complex than the natural images ResNet was designed for. While the original ResNet18 contains blocks that have 64, 128, 256 and 512 feature channels, we reduce these numbers to 16, 32, 32 and 32 yielding around 120,000 trainable parameters while maintaining the network depth. Thus it matches the 142,000 parameters of hCNN [1]. We refer to this model as "narrow ResNet18," abbreviated by nRN18. Additionally, in some experiments we use an extremely reduced network with channel numbers of 8, 16, 16 and 16, called super narrow Resnet18 (snRN18). This network has only around 30,000 parameters.

2.3 Implementation

If not described otherwise, we use the Adam optimizer [11] to minimize the binary cross entropy loss with a learning rate of 0.001 and a batch size of 196. Except for the analysis of sample efficiency

[1] personal communication with the author [1]

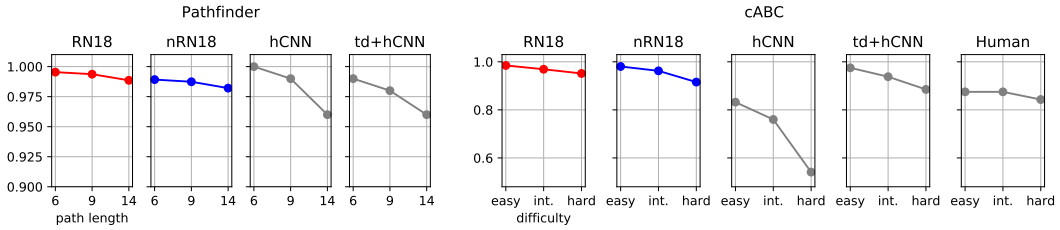

Figure 2: Accuracies of RN18 (red) and nRN18 (blue) on Pathfinder (left) and cABC (right). Scores in gray are adapted from [2].

(Fig. 6), we train our models on 360,000 samples for Pathfinder and 144,000 samples for cABC. We train for up to 80 epochs with an early stopping criterion after 7 epochs without improvement on the validation loss. Images are scaled to 150x150 pixels. We use the PyTorch [12] framework.

2.4 Results

Our main result is that both datasets – Pathfinder and cABC – can be solved with the same accuracy as models including recurrence and feedback [2] using a plain ResNet-18 (Fig. 2, red) or even a more parameter-efficient narrow ResNet-18 (Fig. 2, blue). On Pathfinder, nRN18 matches the performance of hCNN's best-of-5 run (Fig. 2, gray). On cABC, nRN18 even outperforms the state-of-the-art network td+hCNN [2] (hCNN fails on this task). Due to the different training set sizes, this comparison is not entirely fair, but demonstrates that ResNet can solve the benchmark very well given enough training samples.

These experiments show that feedback or recurrent connections are not required to solve the proposed tasks. In addition, feedforward CNNs do not need more parameters to solve the tasks. nRN18 has approximately 120,000 parameters. The hCNN model used in [2] has slightly more parameters (144,000), while td+hCNN is substantially more complex (725,000 parameters). In fact, in section 2.4.1 we will see that we can use an even smaller model without sacrificing a lot of accuracy.

Stability In all other experiments we provide only point estimates of the accuracy rather than intervals due to computational demands. To assess how reliable these results are, we train nRN18 on Pathfinder-14 ten times. We make three observations (Fig. 3):

- Larger batch sizes are more stable.
- The larger RN18 performs better on cABC and is generally more stable than nRN18.
- Pathfinder seems to be more sensitive to batch size but in general easier to solve. The latter is suggested by a generally larger fraction of correctly classified samples in Pathfinder and a smaller gap between RN18 and nRN18.

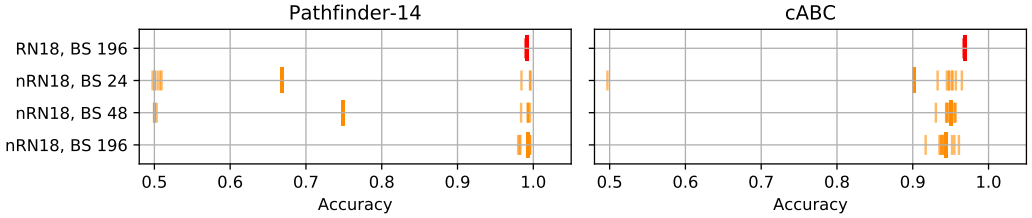

Figure 3: Training stability for RN18 and nRN18 for different batch sizes.

Training/Inference Speed We found our models to train magnitudes faster than the public hCNN implementation. The reason for computational complexity is that each hCNN-GRU cell carries out two convolutions with a filter size of 15x15 with stride 1 on a 150x150 input grid. Although the number of channels is fairly small (20), these convolutions in each timestep require approximately 3.3 billion multiplications. The td+hCNN model, which is more complex than hCNN, uses 8 timesteps.

	RN18	nRN18	hGRU (15x15)
Training (360K 150x150 images)	4.4 min	3.0 min	475 min
Forward (8 × 8 samples)	0.64s	0.23s	34.2s

Table 1: Approximate times required for training (measured on GPU) and forward passes (measured on CPU) on Pathfinder.

Training Issues We found that small batch sizes (12 for RN18 and nRN18, 24 only for nRN18) can prevent the training on Pathfinder-14 from converging. Considering this in conjunction with the finding that small batch sizes tend to be unstable (Fig. 3), we recommend using large batch sizes. In addition, training progresses slowly at the beginning (Fig. 4). It might take more than seven runs over the entire dataset before the training loss begins to decrease significantly. Thus early stopping criteria must be chosen carefully.

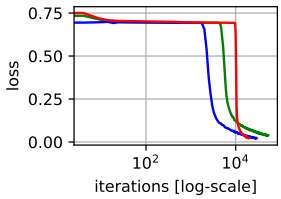

Figure 4: Training loss of snRN18 (green), nRN18 (blue) and RN18 (red) on Pathfinder-14.

2.4.1 Model Complexity

A key claim of [1, 2] is that recurrence enables parameter-efficient models. Hence it seems natural to ask: How many parameters are actually needed by our ResNets? We find that by reducing the number of channels in each layer, we can train ResNet variants with smaller number of parameters than hCNN and td+hCNN to the same level of accuracy (Fig. 5, left).

Interestingly, the smaller the number of channels and thus parameters in a model, the more iterations are required for convergence (Table 2 and Fig. 5 right). This phenomenon can be explained with the lottery ticket hypothesis [13–15], which states that by randomly initializing the weights, sub-networks capable of partially solving the tasks emerge. As the chances of accidentally initializing a well-performing sub-network increase with model size, small models may have to learn more while larger models can rely on identifying a well-initialized sub-network.

		# iterations		Acc	
	params.	PF	cABC	PF	cABC
snRN18	0.03M	53K	22K	97.3	89.3
nRN18	0.12M	29K	10K	98.1	91.9
RN18	11.17M	20K	7K	98.7	95.4

Table 2: Model complexity, training iterations and accuracies for ResNets: Smaller models require more iterations to train.

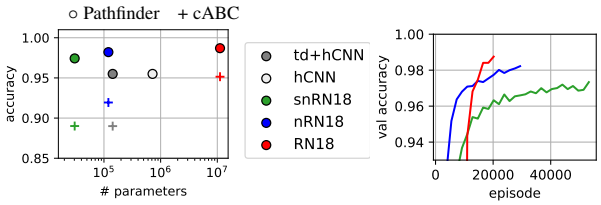

Figure 5: Parameter-performance trade-off (left, accuracy for hCNN on cABC is ∼54%) and validation accuracy during training on Pathfinder-14 (right) for different models.

2.4.2 Sample Efficiency

Sample efficiency expresses which performance (here in terms of accuracy) can be achieved with a fixed number of training examples, i.e. it measures how efficient a model makes use of the samples it sees. Choosing the right inductive biases, e.g. shift equivariance of convolutional layer for many image processing tasks, often improves sample efficiency.

A key argument of Kim et al. [2] is the high sample efficiency of their models as they use only 60,000 (Pathfinder) and 40,000 (cABC) samples. Hence, it seems natural to assess the sample efficiency of our ResNets models, too. To this end, we train each model for 25,000 iterations with validation error computation (on 5,000 samples) every 100 iterations and early stopping with patience 20 (number of validation runs without improvement after which the training is stopped).

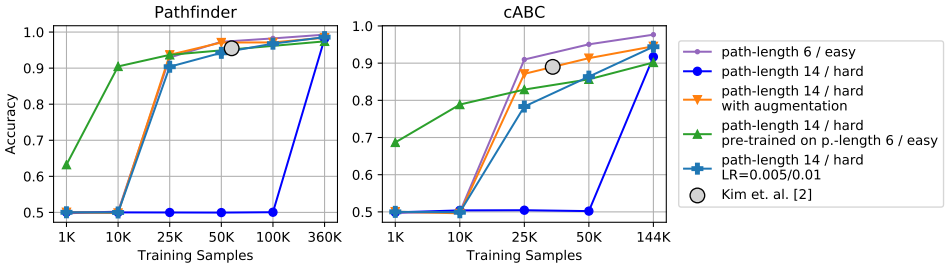

Figure 6: Classification accuracy for different training dataset sizes using nRN18.

First, we observe that the easy benchmark settings require fewer samples than the hard ones. In case of hard samples, nRN18 indeed seems to be more data-hungry than td+hCNN [2]. Thus, at first glance, these results seem to confirm the superior sample efficiency of the recurrent networks.

However, we found three simple ways to improve the sample efficiency of nRN18 and bring it on par with its recurrent competitors: (1) using a larger learning rate, (2) using a simple form of data augmentation by randomly flipping images along their horizontal and vertical axes (as done in [1]) and (3) using a form of curriculum learning by initializing the weights from a model trained on the easiest version of the respective dataset (Pathfinder: path-length 6; cABC: easy).[2]

Using a larger learning rate (0.005 for Pathfinder, 0.01 for cABC) improves sample efficiency substantially (at the price of training stability[3]). While this finding might seem surprising, it is consistent with previous research on batch normalization (BN): While Santurkar et al. [16] found BN to facilitate the optimization by smoothing both loss landscape and gradient, Bjorck et al. [17] argue the effects of improved generalization and faster convergence are due to the larger learning rates enabled by BN.

Furthermore, both other strategies to increase sample efficiency – data augmentation and pre-training – work well for nRN18, too (Fig. 6). The latter is particularly remarkable as it allows for very small training sets and shows that sample efficiency is to some degree a matter of choosing the right initializations. By contrast, the effect of augmentation sets in only above 25,000 samples.

3 Conclusion

Our work shows that the Pathfinder and cABC benchmarks can be solved using parameter-effient and sample-efficient feedforward convolutional networks, contrary to previous belief. There is no fundamental limitation that prevents CNNs from following paths through learning long-range dependencies, although training can be challenging due to instability of small batch sizes. We were able to use ResNets with only 30,000 parameters and found that a high sample efficiency can be attained through various means: large learning-rates, augmentation or pre-training. The fact that our CNN architectures are identical in all experiments contradicts the claim by Kim et al. [2] that feedforward models must be specifically tuned for complex visual tasks. Despite these findings, we do believe that recurrence could be an essential ingredient for intelligent systems, particularly in tasks requiring a high degree of processing depth.

[2]For the pre-trained models, the validation interval is set to 50 instead of 100 iterations due to fast convergence.

[3]The reported numbers for Pathfinder at lr=0.005 for 100K and 360K samples were obtained in a second run after the first round of training did not converge. cABC uses a batch size of 512 at 144K samples. Note, Kim et al. [2] report best-of-five scores.

References

[1] Drew Linsley, Junkyung Kim, Vijay Veerabadran, Charles Windolf, and Thomas Serre. Learning long-range spatial dependencies with horizontal gated recurrent units. In *Advances in neural information processing systems*, pages 152–164, 2018.

[2] Junkyung Kim, Drew Linsley, Kalpit Thakkar, and Thomas Serre. Disentangling neural mechanisms for perceptual grouping. *ICLR*, 2019.

[3] Drew Linsley, Junkyung Kim, and David Berson. Sample-efficient image segmentation through recurrence. *arXiv: Computer Vision and Pattern Recognition*, 2018.

[4] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[5] Kyunghyun Cho, Bart van Merrienboer, Çaglar Gülçehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. In *EMNLP*, 2014.

[6] Shaojie Bai, J. Zico Kolter, and Vladlen Koltun. An empirical evaluation of generic convolutional and recurrent networks for sequence modeling. *arXiv:1803.01271*, 2018.

[7] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems (NIPS)*, pages 5998–6008, 2017.

[8] T. Kulvicius, S. Herzog, T. Lüddecke, M. Tamosiunaite, and F. Wörgötter. One-shot multi-path planning for robotic applications using fully convolutional networks. In *In Proceedings of International Conference on Robotics and Automation (ICRA)*, 2020.

[9] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[10] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 248–255. IEEE, 2009.

[11] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2014.

[12] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. In *Advances in Neural Information Processing Systems Workshops*, 2017.

[13] Jonathan Frankle and Michael Carbin. The lottery ticket hypothesis: Finding sparse, trainable neural networks. *ICLR*, 2019.

[14] Hattie Zhou, Janice Lan, Rosanne Liu, and Jason Yosinski. Deconstructing lottery tickets: Zeros, signs, and the supermask. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 3597–3607. Curran Associates, Inc., 2019. URL `http://papers.nips.cc/paper/8618-deconstructing-lottery-tickets-zeros-signs-and-the-supermask.pdf`.

[15] Jonathan Frankle, Gintare Karolina Dziugaite, Daniel M Roy, and Michael Carbin. Linear mode connectivity and the lottery ticket hypothesis. *ICML*, 2020.

[16] Shibani Santurkar, Dimitris Tsipras, Andrew Ilyas, and Aleksander Madry. How does batch normalization help optimization? In *Advances in Neural Information Processing Systems*, pages 2483–2493, 2018.

[17] Nils Bjorck, Carla P Gomes, Bart Selman, and Kilian Q Weinberger. Understanding batch normalization. In *Advances in Neural Information Processing Systems*, pages 7694–7705, 2018.