# TANDEM BLOCKS IN DEEP CONVOLUTIONAL NEURAL NETWORKS

**Anonymous authors**
Paper under double-blind review

## ABSTRACT

Due to the success of residual networks (ResNets) and related architectures, shortcut connections have quickly become standard tools for building convolutional neural networks. The explanations in the literature for the apparent effectiveness of shortcuts are varied and often contradictory. We hypothesize that shortcuts work primarily because they act as linear counterparts to nonlinear layers. We test this hypothesis by using several variations on the standard residual block, with different types of linear connections, to build small (100k–1.2M parameter) image classification networks. Our experiments show that other kinds of linear connections can be even more effective than the identity shortcuts. Our results also suggest that the best type of linear connection for a given application may depend on both network width and depth.

## 1 INTRODUCTION

Deep convolutional neural networks have become the dominant force for many image classification tasks; see Krizhevsky et al. (2012); Simonyan & Zisserman (2014); Szegedy et al. (2014). Their ability to assimilate low-, medium-, and high-level features in an end-to-end multi-layer fashion has led to myriad groundbreaking advances in the field. In recent years, residual networks (ResNets) have emerged as one of the best performing neural network archetypes in the literature; see He et al. (2015). Through the use of identity shortcut connections, ResNets have overcome the challenging technical obstacles of vanishing/exploding gradients and the apparent degradation that otherwise comes with training very deep networks. ResNets have achieved state-of-the-art performance on several image classification datasets using very deep neural networks, sometimes with over 1000 layers.

Although shortcut connections appeared in the early neural network literature, e.g., Bishop (1995); Ripley & Hjort (1995); Schraudolph (2012), their importance became more clear in 2015 with the emergence of the HighwayNets of Srivastava et al. (2015) and ResNets. The former involved gated shortcut connections that regulate the flow of information across the network, while the latter used identity shortcut connections, which are parameterless. ResNets are also presumed to be easier to train and seem to perform better in practice. In their first ResNet paper, He et al. argued that identity maps let gradients flow back, enabling the training of very deep networks, and that it's easier for a layer to learn from a starting point of keeping things the same (the identity map) than from the zero map; see also He et al. (2016).

However, in a flurry of recent activity, most notably from Zagoruyko & Komodakis (2016); Greff et al. (2016); Veit et al. (2016); Li et al. (2016) and Wu et al. (2016), arguments emerged that the effectiveness of ResNets was not really due to their depth, where practitioners were training networks of hundreds or thousands of layers, but rather that deep ResNets were effectively creating ensembles of shallower networks, and the layers were much more likely to refine and reinforce existing features than engineer new ones. These arguments effectively assert that the achievement of ResNets is less about extreme depth and more about their ability to ease backpropegation with moderate depth. Indeed in many cases wider residual networks that were only moderately deep (10-50 layers) were shown to perform as well, and in less time than very deep ones (over 100 layers).

More recently still, others have presented many clever and creative ways to train very deep networks using variations on the shortcut theme; see for example Huang et al. (2016); Larsson et al. (2016); Zhang et al. (2016a); Han et al. (2016); Abdi & Nahavandi (2016); Chen et al. (2017); Zhang et al.

(2016b); He et al. (2015); Lee et al. (2017); Xie et al. (2016); Savarese (2016); Szegedy et al. (2014; 2015), and Szegedy et al. (2016). In summary, shortcut connections clearly help in practice, but there are many different, and sometimes conflicting hypotheses as to why.

In this paper we investigate a new hypothesis about shortcut connections, namely, that their power lies not in the identity mapping itself, but rather just in combining linear and nonlinear features at each layer. We first describe some of the intuition about why this might be the case. We then investigate this idea with careful experiments using relatively small networks constructed of five different types of blocks. These blocks are all variations on the idea of residual blocks (ResBlocks), but where the identity shortcut is replaced with a more general linear function. We call these blocks, consisting of both a linear and a nonlinear part, *tandem blocks* and the resulting networks *tandem networks*. Residual networks and several similar architectures can be seen as special cases of tandem networks.

The networks we use in our experiments are relatively small (100k–1.2M parameter) image classification networks constructed from these various tandem blocks. The small networks are appropriate because the goal of the experiments is not to challenge state-of-the-art results produced by much larger models, but rather to compare the five architectures in a variety of settings in order to gain insight into their relative strengths and weaknesses.

Our experiments suggest that more general linear shortcuts tend to outperform the identity shortcut of ResNets, and that the best type of linear connection to use in the blocks of a tandem network depends on several factors, including both network width and depth.

## 2 Why Tandem Blocks?

The basic building block we use is the *tandem block*, which consists of a linear and a nonlinear part in parallel (see Figures 1 and 2). These are analogous to ResNet blocks where the identity shortcut has been replaced with a more general linear function. The outputs of the two parts is summed and then passed to subsequent blocks.

Note that, unlike in the case of the original ResNet paper, the resulting sum is not passed to another nonlinear activation function. This is because recent work of Dong et al. (2017) has shown that removing the activation function after the sum in ResNets improves performance.

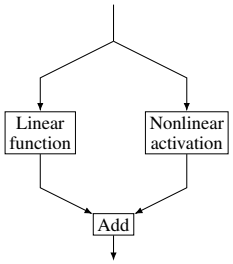



Figure 1: Illustrating the basic idea of a tandem block, replacing the identity in ResNets with a linear transformation, and adding the two activation functions together. Note that, unlike He et al. (2015), we do not use an activation function after the final sum; this is motivated by the results of Dong et al. (2017).

There are at least three reasons to hypothesize that network blocks with both a linear and a nonlinear component running parallel to each other (like ResBlocks), should perform well in deep networks.

(i) Linear functions behave very well under backpropagation, not only are they simple to calculate, but they also do not cause gradients to vanish, unless they learn weights that badly disproportionate (but this can be handled by appropriate initialization). Therefore, removing some nonlinear activation functions to make parts of a block purely linear should facilitate training deep networks.

(ii) The tandem setup, with both linear and nonlinear parts in each block, gives paths through the network with varying degrees of nonlinearity. This should give a final network that

behaves something like an ensemble of regular (fully nonlinear) subnetworks of many depths. Going deeper should have little or no cost because it is essentially equivalent to adding some additional subnetworks to the ensemble. In fact, it should actually improve performance because the portions corresponding to deeper subnetworks can handle more nuanced patterns while the parts corresponding to shallower subnetworks paint the broad strokes. Thus, the tandem architecture should allow networks to learn optimal levels of complexity easily.

(iii) In their wonderful paper, Dong et al. (2017) show that many network architectures have far too many activation functions, and that performance can be improved by removing activations that can't be avoided via linear paths. Intuitively, this should be more compelling the deeper you go.

Despite these heuristic reasons to expect better deep-network performance from our tandem architectures, they do still risk overfitting; but, as usual, this can be partially handled with standard regularization and dropout techniques.

Of course, one would expect these effects to play out differently in narrow, deep networks versus wide, shallow ones. The rest of this paper is focused on some specific careful experiments to explore and test these hypotheses.

## 3 Experiment Design

We experiment with several different types of basic building blocks and assemble these into several network architectures.

### 3.1 Building Blocks

We propose five different tandem blocks, each of which can be viewed as a variant on the standard ResBlock. We consider blocks whose nonlinear part consists of either one or two layers of activated (nonlinear) convolutions (on the right side in Figure 1). On the linear side of our blocks (the left side in Figure 1) we consider either (i) identity maps, corresponding to standard ResBlocks and (ii) convolutions of size $1 \times 1$ or $3 \times 3$, corresponding to more general tandem blocks. Note that in some cases identity maps are impossible to use because of mismatches in layer width. In those cases, as is typical with ResNets, we use $1 \times 1$ convolutions to project the identity to have the necessary width. In all cases the outputs of the linear nad nonlinear parts are added together at the end.

Specifically, the block variants we defined are as follows:

- $B_{\text{id}}(2, w)$ is the standard residual block, with two activated convolutional layers and an identity connection from start to finish. When required to connect layers of different widths, the identity connection is replaced with a more flexible $1 \times 1$ convolution.
- $B_{\text{id}}(1, w)$ is the same as $B_{\text{id}}(2, w)$, but with only one activated convolution instead of two. This is another form of ResBlock.
- $B_{1\times1}(2, w)$ is a tandem block that is similar to $B_{\text{id}}(2, w)$ except that it always uses $1 \times 1$ convolutions, even when connecting layers of the same width.
- $B_{1\times1}(1, w)$ is a tandem block like $B_{1\times1}(2, w)$, but with only one activated convolution instead of two.
- $B_{3\times3}(1, w)$ has $3 \times 3$ convolutions on both the linear and nonlinear sides, but on the nonlinear side it is followed by a nonlinear activation function while the linear side is not.

These are all shown in Figure 2.

Reasonable heuristic arguments could be made to justify all sorts of expectations for the different blocks. For example, one might expect the identity blocks to perform well in deeper networks because they avoid changes in gradient magnitude, but they might also be less effective than blocks that can use $1 \times 1$ convolutions to recombine features in potentially more useful ways Using $3 \times 3$ convolutions for the linear connections and letting both sides engineer new filters could create more robust networks that need less depth, but it could also be an inefficient use of parameters. Our experiments were designed to find out which of these intuitions are correct.

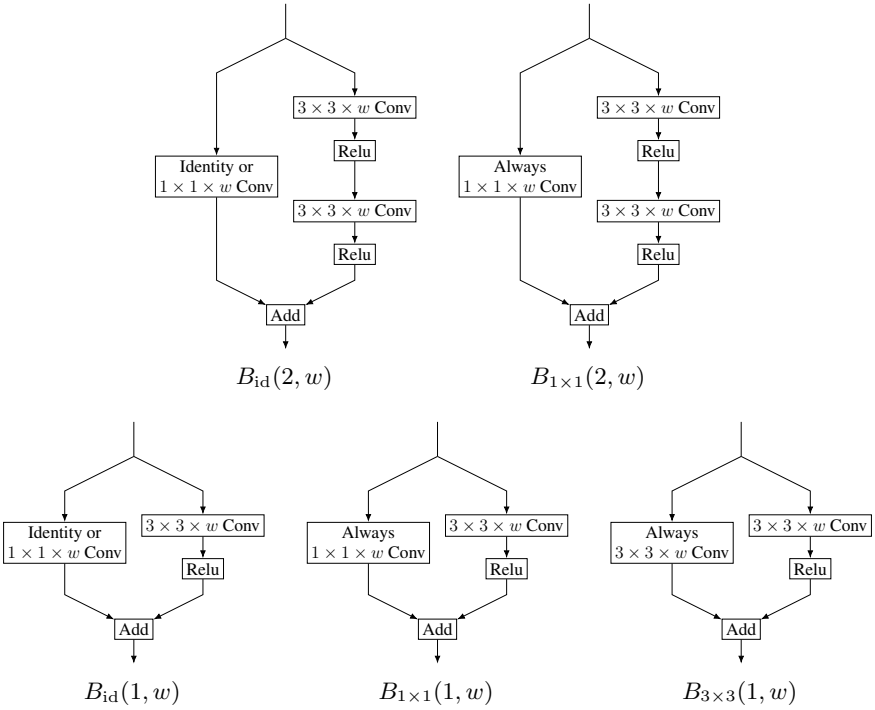



Figure 2: The five tandem blocks used in all of our experiments. The two left-most blocks $B_{\mathrm{id}}(2, w)$ and $B_{\mathrm{id}}(1, w)$ correspond to traditional ResNets and the others to more general tandem nets.

### 3.2 Network Architectures

For our experiments, we focused on small architectures ranging from 8 to 26 layers and from 100k to 1.2M parameters. This was appropriate because the goal of these experiments was not to challenge state-of-the-art results produced by much larger models, but rather to compare the aforementioned five architectures in a variety of settings so as to gain insight into their relative strengths and weaknesses.

Each experiment features five models, corresponding to the five types of block. To ensure fair comparisons, each model has the same number of layers and nearly the same number of parameters. To achieve the latter, different values of the width parameter $w$ must be selected for different blocks. In particular, $B_{3\times3}$ requires significantly smaller values for $w$ than the other blocks because its linear convolutions have so many more parameters.

We also explored different values of hyperparameters for different architectures to make sure that each was achieving its best performance.

### 3.3 Network Shape

For all models, we used a simple architecture with three parameters. The block layer parameter $l$ sets the number of layers in each block, as in Figure 2. The depth parameter $d$ controlls the depth in the network by determining how many times to repeat each block. The width parameter $w_0$ sets the width of each block and is used to control the number of parameters in each model. The architecture is illustrated in Table 1. In every case, the resulting network had $6d + 2$ layers.

### 3.4 Regularization

In each model, we used both dropout and $L^2$ regularization (weight decay). In all blocks, we applied dropout immediately after the linear and nonlinear sides were added together. In blocks with two nonlinear layers, we also applied dropout after the first nonlinear layer. We found that a dropout proportion of .1 was appropriate for all of the models.

| Component | Repetitions |
|---|---|
| Input | 1 |
| $3 \times 3 \times w_0$ Conv | 1 |
| $B(l, w_0)$ | $2d/l$ |
| $B(l, 2w_0)$ with Stride 2 | 1 |
| $B(l, 2w_0)$ | $2d/l - 1$ |
| $B(l, 4w_0)$ with Stride 2 | 1 |
| $B(l, 4w_0)$ | $2d/l - 1$ |
| Global Average Pooling | 1 |
| Softmax Output Layer | 1 |

Table 1: All of the networks used in our experiments were instances of this meta-architecture. The parameters $w_0$, $d$ and $l$ are chosen so that the total depth of each network is the same and the total number of parameters is comparable.

We applied $L^2$ regularization to the weights of every convolution, both linear and nonlinear, but not to the biases. We found that the ResNets $B_{\text{id}}(2, w)$ and $B_{\text{id}}(2, w)$ performed best with a regularization constant of 0.0003; whereas the other tandem nets $B_{1\times1}(2, w)$, $B_{1\times1}(1, w)$, and $B_{3\times3}(1, w)$ all performed best with a regularization constant of 0.0001. While light regularization did improve test accuracy for all five architectures, the improvements were modest and did not significantly change the relative performances of different architectures.

We also used a simple augmentation scheme for the training data: shifting images both horizontally and vertically by a factor of no more than 10% and flipping the images horizontally.

### 3.5 Batch Normalization

We tried inserting batch normalization layers in several places in each block—before activations, after activations, before convolutions, after the addition—and were surprised to find that none of these approaches was helpful. Contrary to the claims usually associated with this batch normalization, we observed that networks with batch normalization achieved about the same performance on average, but were less stable and more sensitive to learning rates. Even in cases where networks with batch normalization slightly outperformed their counterparts without, it took so much longer to train them that it did not seem worth the extra expense and complexity. Accordingly, we left batch normalization out of all of our experiments.

### 3.6 Initialization

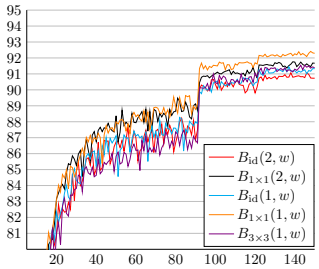
We initialized all of the weights, but not the biases, by sampling from a truncated normal distribution (cut off at two standard deviations) with zero mean. Standard deviations were scaled from a base value by fan-in, as in He et al. (2015). The appropriate base standard deviation varied considerably from network to network. Some were as high as 1.0 while others were as low as 0.01. Aside from this variation, the responses to different base standard deviations were consistent. Low values produced networks that didn't learn, while high values produced networks that diverged.

This behavior suggests that all of these networks might all have loss functions with similar broad characteristics, including perhaps a nonoptimal neighborhood near zero with small gradient, thus gradient descent makes little or no progress in this region. Farther from the origin, the loss functions may have many local optima and a gradient of high variance, causing gradient descent to bounce around, unproductively, even with a very low learning rate.

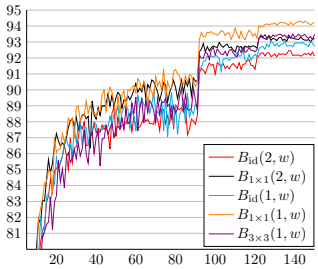
### 3.7 Learning Rate and Descent Method

We used the Adam method of gradient descent for all experiments, with the authors' recommended hyperparameters; see Kingma & Ba (2014). Even with adaptive learning rates, we found that a learning rate schedule significantly improved results across the board. For CIFAR-10 and CIFAR-100, we used a low learning rate of 0.0002 for the first epochs, then increased it to 0.001 through

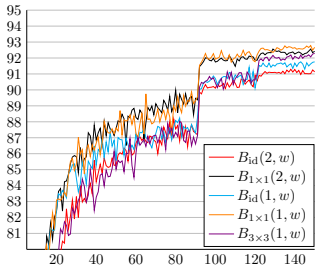
epoch 90, decreased again to 0.0002 through epoch 120, and finally to 0.00004 until epoch 150. For SVHN and Fashion-MNIST, we scaled this schedule to 100 epoch runs.



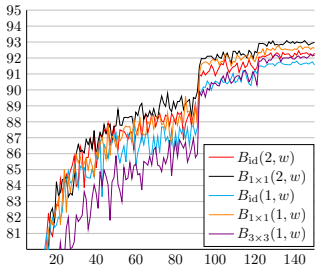

14 Layers, 300k Parameters




14 Layers, 1.2m Parameters




20 Layers, 475k Parameters




26 Layers, 640k Parameters

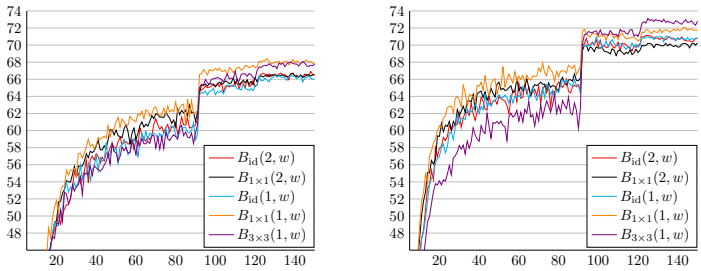
Figure 3: Plots of test accuracy vs epoch for all CIFAR-10 experiments. In each case, the two tandem models $B_{1\times1}(1, w)$ and $B_{1\times1}(2, w)$ consistently beat the two ResNet models $B_{\text{id}}(1, w)$ and $B_{\text{id}}(2, w)$.

| Dataset | Layers | Params | $B_{\text{id}}(2)$ | $B_{\text{id}}(1)$ | $B_{1\times1}(2)$ | $B_{1\times1}(1)$ | $B_{3\times3}(1)$ |
|---|---|---|---|---|---|---|---|
| CIFAR-10 | 14 | 300k | 91.07 | 91.44 | 91.76 | **92.41** | 91.54 |
| CIFAR-10 | 14 | 1.2m | 92.46 | 93.05 | 93.39 | **94.31** | 93.50 |
| CIFAR-10 | 20 | 470k | 91.29 | 91.76 | 92.56 | **92.78** | 92.29 |
| CIFAR-10 | 26 | 640k | 92.45 | 91.80 | **93.09** | 92.75 | 92.22 |
| CIFAR-100 | 14 | 300k | 66.92 | 66.48 | 66.68 | **68.42** | 68.08 |
| CIFAR-100 | 14 | 1.2m | 71.14 | 71.05 | 70.27 | 72.06 | **73.12** |
| CIFAR-100 | 20 | 470k | 66.44 | 65.27 | 67.81 | **68.33** | 65.96 |
| CIFAR-100 | 26 | 640k | 68.64 | 65.31 | 68.53 | **69.20** | 66.83 |
| SVHN | 8 | 130k | 95.25 | 95.01 | **95.61** | 95.39 | 95.16 |
| SVHN | 14 | 300k | 96.24 | 96.21 | 96.49 | **96.58** | 96.55 |
| SVHN | 20 | 470k | 96.43 | 96.38 | 96.87 | **96.89** | 96.76 |
| Fashion-MNIST | 8 | 130k | 89.66 | 90.25 | 91.63 | **92.01** | 91.69 |
| Fashion-MNIST | 14 | 300k | 91.94 | 92.38 | 92.95 | **93.94** | 93.84 |

Table 2: The highest test accuracy achieved with each tandem block in each experiment. Note that the standard residual blocks $B_{\text{id}}(2)$ and $B_{\text{id}}(1)$ were always outperformed by the others.

## 4 Results

To evaluate the five blocks, we used each to build several different networks and then tested them on four popular image recognition problems: CIFAR-10, CIFAR-100, SVHN, and Fashion-MNIST. In total, we made the five blocks compete in thirteen challenges. The best blocks in each challenge



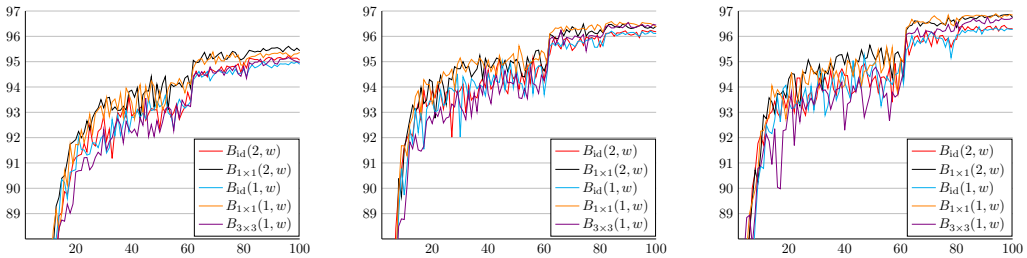

14 Layers, 300k Parameters

14 Layers, 1.2m Parameters

20 Layers, 475k Parameters

26 Layers, 640k Parameters

Figure 4: Plots of test accuracy vs epoch for all CIFAR-100 experiments. For both 14-layer networks (top), the two winners were the tandem models $B_{1\times1}(1, w)$ and $B_{3\times3}(1, w)$. Both ResNet models $B_{\mathrm{id}}(1, w)$ and $B_{\mathrm{id}}(2, w)$ performed worse, along with $B_{1\times1}(2, w)$. For the deeper models (bottom), the underperforming models were $B_{3\times3}(1, w)$ and the ResNet model $B_{\mathrm{id}}(1, w)$, whereas the better performing models were the tandem models $B_{1\times1}(1, w)$ and $B_{1\times1}(2, w)$, as well as the ResNet model $B_{\mathrm{id}}(2, w)$




8 Layers, 130k Parameters

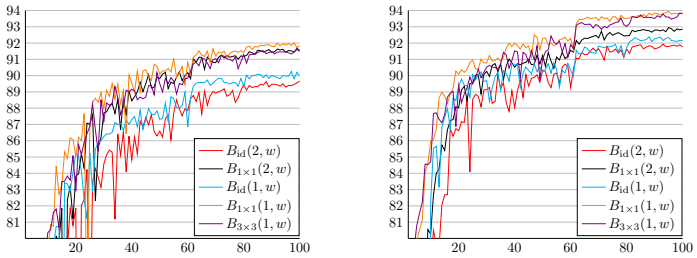
14 Layers, 300k Parameters

20 Layers, 475k Parameters

Figure 5: Plots of test accuracy vs epoch for all SVHN experiments. In all three cases the tandem models $B_{1\times1}(1, w)$ and $B_{1\times1}(2, w)$ outperformed the ResNet models $B_{\mathrm{id}}(1, w)$ and $B_{\mathrm{id}}(2, w)$. The $3 \times 3$ model $B_{3\times3}(1, w)$ was in two cases competitive with the other tandem models and less so in the other case.

were competitive with the best published results for their numbers of parameters; see Table 2 for the breakdown.

In all four challenges on CIFAR-10, the two $B_{1\times1}$ variants (which always use learnable projections instead of identity maps) excelled while the $B_{\mathrm{id}}$ variants, with their more standard identity connections, lagged behind. As network depth increased the two-layer blocks overtook their one-layer counterparts, suggesting that deeper blocks might be more advantageous in deeper architecture. Though the order of the others changed, $B_{3\times3}$ remained consistently in the middle of the pack.




8 Layers, 130k Parameters

14 Layers, 300k Parameters

Figure 6: Plots of test accuracy vs epoch for all Fashion-MNIST experiments. In both cases the three tandem models outperformed the two ResNet models.

The results for CIFAR-100 were in many ways similar to those for CIFAR-10. The two-layer blocks caught up to their one-layer siblings as depth increased. This was due in part to the fact that the one-layer blocks barely improved as depth increased; $B_{3\times3}$ actually got a bit worse. However, $B_{3\times3}(1, w)$ shined with the shallower architectures, particularly on the wider one with 1.2m parameters. This impressive result suggests that $B_{3\times3}$ might be even better suited to building relatively wide and shallow networks than the Resblocks used to achieve state-of-the-art results in the WideResNets paper.

On SVHN, the tandem nets $B_{1\times1}(2)$ and $B_{1\times1}(1)$ once again excelled while the traditional ResBlocks $B_{\text{id}}(2)$ and $B_{\text{id}}(1)$ stayed slightly behind. Interestingly, $B_{3\times3}(1)$ performed nearly as well as $B_{1\times1}(2)$ and $B_{1\times1}(1)$ in the 14- and 20-layer networks despite their narrow shape. On the Fashion-MNIST dataset, $B_{1\times1}(1)$ and $B_{3\times3}(1)$ had the strongest performance and the ResNets $B_{\text{id}}(2)$ and $B_{\text{id}}(1)$ were significantly behind.

## 5 Conclusions

We generalized residual blocks (which use identity shortcut connections) to tandem blocks (which can use any linear map, not just the identity). We found that general linear shortcuts have the same benefits as the identity maps in residual blocks, and they can actually increase performance compared to identity shortcuts. These results seem to confirm that the success of residual networks and related architectures is not due to special properties of shortcut connections or identity maps, but rather is simply a result of using linear maps to complement nonlinear ones.

The additional flexibility gained by replacing identity maps with $1 \times 1$ or even $3 \times 3$ convolutions led to better results in every one of our experiments. This was not due to extra parameters, as we adjusted layer widths to keep parameter counts as close to equal as possible. Instead, it appears that linear convolutions do a better job than identity maps of working together with nonlinear convolutions.

Our results suggest that the best type of tandem block for a given application depends on the shape of the network being built. Blocks with one nonlinear convolution do better in shallower networks while blocks with two do better in deeper ones. Blocks that use $3 \times 3$ convolutions for their linear connections are effective in general and may be even better in wide networks than those with $1 \times 1$s.

Finally, we note that there are many more possible types of tandem block than we consider here, and many more applications in which to test them.

### Acknowledgments

Removed for review.

## REFERENCES


Masoud Abdi and Saeid Nahavandi. Multi-residual networks. *CoRR*, abs/1609.05672, 2016. URL http://arxiv.org/abs/1609.05672.

Christopher M. Bishop. *Neural Networks for Pattern Recognition*. Oxford University Press, Inc., New York, NY, USA, 1995. ISBN 0198538642.

Yunpeng Chen, Jianan Li, Huaxin Xiao, Xiaojie Jin, Shuicheng Yan, and Jiashi Feng. Dual path networks. *CoRR*, abs/1707.01629, 2017. URL http://arxiv.org/abs/1707.01629.

Xuanyi Dong, Guoliang Kang, Kun Zhan, and Yi Yang. Eraserelu: A simple way to ease the training of deep convolution neural networks. *CoRR*, abs/1709.07634, 2017. URL http://arxiv.org/abs/1709.07634.

Klaus Greff, Rupesh Kumar Srivastava, and Jürgen Schmidhuber. Highway and residual networks learn unrolled iterative estimation. *CoRR*, abs/1612.07771, 2016. URL http://arxiv.org/abs/1612.07771.

Dongyoon Han, Jiwhan Kim, and Junmo Kim. Deep pyramidal residual networks. *CoRR*, abs/1610.02915, 2016. URL http://arxiv.org/abs/1610.02915.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015. URL http://arxiv.org/abs/1512.03385.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Identity mappings in deep residual networks. *CoRR*, abs/1603.05027, 2016. URL http://arxiv.org/abs/1603.05027.

Gao Huang, Zhuang Liu, and Kilian Q. Weinberger. Densely connected convolutional networks. *CoRR*, abs/1608.06993, 2016. URL http://arxiv.org/abs/1608.06993.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2014. URL http://arxiv.org/abs/1412.6980.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger (eds.), *Advances in Neural Information Processing Systems 25*, pp. 1097–1105. Curran Associates, Inc., 2012.

Gustav Larsson, Michael Maire, and Gregory Shakhnarovich. Fractalnet: Ultra-deep neural networks without residuals. *CoRR*, abs/1605.07648, 2016. URL http://arxiv.org/abs/1605.07648.

Youngwan Lee, Huieun Kim, Eunsoo Park, Xuenan Cui, and Hakil Kim. Wide-residual-inception networks for real-time object detection. *CoRR*, abs/1702.01243, 2017. URL http://arxiv.org/abs/1702.01243.

Sihan Li, Jiantao Jiao, Yanjun Han, and Tsachy Weissman. Demystifying resnet. *CoRR*, abs/1611.01186, 2016. URL http://arxiv.org/abs/1611.01186.

Brian D. Ripley and N. L. Hjort. *Pattern Recognition and Neural Networks*. Cambridge University Press, New York, NY, USA, 1st edition, 1995. ISBN 0521460867.

Pedro H. P. Savarese. Learning identity mappings with residual gates. *CoRR*, abs/1611.01260, 2016. URL http://arxiv.org/abs/1611.01260.

Nicol N. Schraudolph. Centering neural network gradient factors. In *Neural Networks: Tricks of the Trade (2nd ed.)*, volume 7700 of *Lecture Notes in Computer Science*, pp. 205–223. Springer, 2012.

Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556, 2014. URL http://arxiv.org/abs/1409.1556.

Rupesh Kumar Srivastava, Klaus Greff, and Jürgen Schmidhuber. Highway networks. *CoRR*, abs/1505.00387, 2015. URL http://arxiv.org/abs/1505.00387.

Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott E. Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. *CoRR*, abs/1409.4842, 2014. URL http://arxiv.org/abs/1409.4842.

Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jonathon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. *CoRR*, abs/1512.00567, 2015. URL http://arxiv.org/abs/1512.00567.

Christian Szegedy, Sergey Ioffe, and Vincent Vanhoucke. Inception-v4, inception-resnet and the impact of residual connections on learning. *CoRR*, abs/1602.07261, 2016. URL http://arxiv.org/abs/1602.07261.

Andreas Veit, Michael J. Wilber, and Serge J. Belongie. Residual networks are exponential ensembles of relatively shallow networks. *CoRR*, abs/1605.06431, 2016. URL http://arxiv.org/abs/1605.06431.

Zifeng Wu, Chunhua Shen, and Anton van den Hengel. Wider or deeper: Revisiting the resnet model for visual recognition. *CoRR*, abs/1611.10080, 2016. URL http://arxiv.org/abs/1611.10080.

Saining Xie, Ross B. Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He. Aggregated residual transformations for deep neural networks. *CoRR*, abs/1611.05431, 2016. URL http://arxiv.org/abs/1611.05431.

Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. *CoRR*, abs/1605.07146, 2016. URL http://arxiv.org/abs/1605.07146.

Ke Zhang, Miao Sun, Tony X. Han, Xingfang Yuan, Liru Guo, and Tao Liu. Residual networks of residual networks: Multilevel residual networks. *CoRR*, abs/1608.02908, 2016a. URL http://arxiv.org/abs/1608.02908.

Xingcheng Zhang, Zhizhong Li, Chen Change Loy, and Dahua Lin. Polynet: A pursuit of structural diversity in very deep networks. *CoRR*, abs/1611.05725, 2016b. URL http://arxiv.org/abs/1611.05725.