ShaResNet: reducing residual network parameter number by sharingweights

Alexandre Boulch

ShaResNet: reducing residual network parameter number by sharing weights

Alexandre Boulch
ONERA, The French Aerospace Lab, F-91761 Palaiseau, France

Abstract

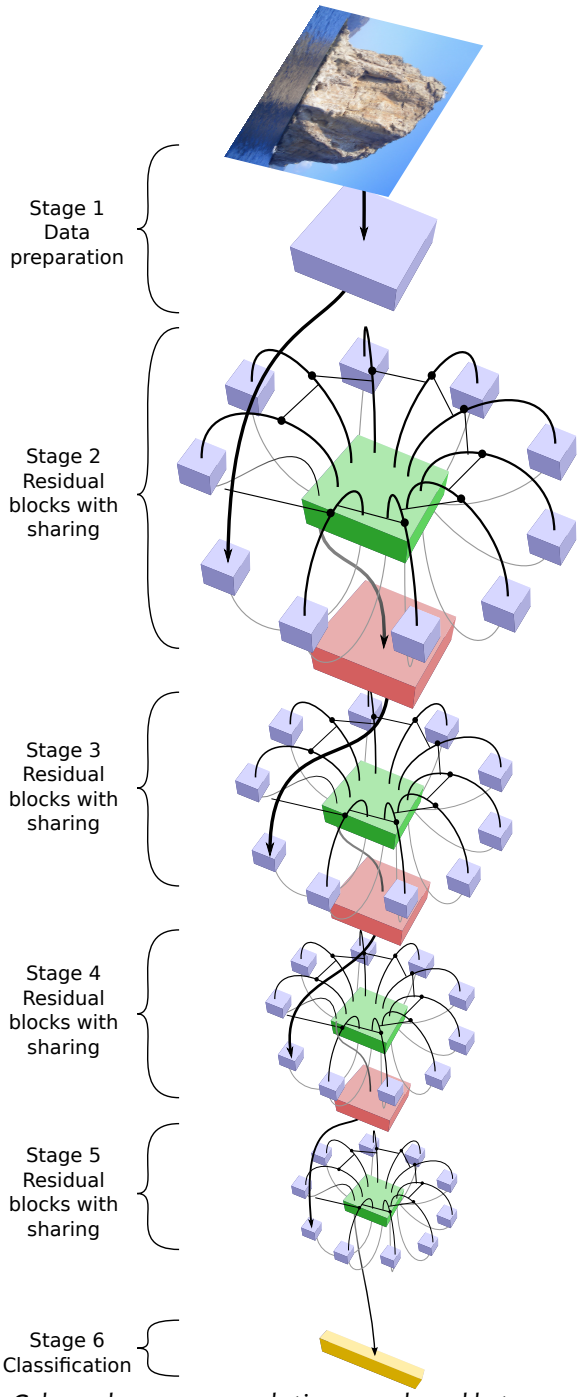
Deep Residual Networks have reached the state of the art in many image processing tasks such image classification. However, the cost for a gain in accuracy in terms of depth and memory is prohibitive as it requires a higher number of residual blocks, up to double the initial value. To tackle this problem, we propose in this paper a way to reduce the redundant information of the networks. We share the weights of convolutional layers between residual blocks operating at the same spatial scale. The signal flows multiple times in the same convolutional layer. The resulting architecture, called ShaResNet, contains block specific layers and shared layers. These ShaResNet are trained exactly in the same fashion as the commonly used residual networks. We show, on the one hand, that they are almost as efficient as their sequential counterparts while involving less parameters, and on the other hand that they are more efficient than a residual network with the same number of parameters. For example, a 152-layer-deep residual network can be reduced to 106 convolutional layers, i.e. a parameter gain of 39%, while loosing less than 0.2% accuracy on ImageNet.

1 Introduction

Convolutional Neural Networks (CNNs) are now widely used for image processing tasks from classification [LBBH98] and object detection [GDDM14, RHGS15] to semantic segmentation [BKC15, ASL16]. Their utilisation even generalizes to other fields where data can be represented as tensors like in point cloud processing [BM16] or 3D shape style identification [LGK16]. Today's network architectures still carry a strong inheritance of the CNN early stage designs. They are based on stacking convolutional, activation and dimensionality reduction layers. Over the past years, the progress in image processing tasks went together with a gradual increase in the number of layers, from AlexNet [KSH12] to Residual networks [HZRS16a] (ResNets) that may contain up to hundreds of convolutions.

Practical use of such networks may be challenging when using low memory system, such as autonomous vehicles, both for optimization and inference. Moreoever, from a biological point of view, a higher number of stacked layers leads to networks further from the original underlying idea of neural networks: biological brain mimicry. According to the current knowledge of the brain, cerebral cortex is composed of a low number of layers where the neurons are highly connected. Moreover the signal is also allowed to recursively go through the same neurons. In that sense recurrent neural networks are much closer to the brain structure but more difficult to optimize [BSF94, HS97].

Looking more closely at the repetition of residual blocks in ResNets, it could somehow be interpreted as an unwrapped recurrent neural networks. This constatation raises questions such as *"how similar are the weights of the blocks ?"*, *"do the same parts of the blocks operate similar operations ?"* and in the later case *"is it possible to reduce the parameter number of a residual network ?"*. Driven by these observations and



Color code: green convolutions are shared between several residual blocks, red blocks are spatial dimensionality reductions (covolution or pooling), blue ones are the other convolutions and the yellow block is the classifier, average pooling or fully connected.

Figure 1: 3D representation of a residual network with shared convolution.

questions, we present a new network architecture based on residual networks where part of the convolutions share weights, called *ShaResNets*. It results in a great decrease of the number of network parameters, from 25% to 45% depending on the size of the original architecture. Our networks also present a better ratio performances over parameter number while downgrading the absolute performance by less than 1%.

The paper is organized as follow: section 2 presents the related work on convolutional neural networks (CNNs) ; the ShaResNets are presented in section 3 and finally, in section 4, we expose our experimentations on classification datasets CIFAR 10 and 100 and ILSVRC Imagenet.

2 Related work

CNNs were introduced in [LBD$^+$89] for hand written digits recognition. They became over the past years one of the most enthusiastic field of deep learning [LBH15]. The CNNs are usually built using a common framework. They contains many convolutional layers and operate a gradual spatial dimension reduction using convolutional strides or pooling layers [CMS12, JKL$^+$09]. This structure naturally integrates low/mid/high level features along with a dimension compression before ending with a classifier, commonly a perceptron [Ros57], multi-layered or not, i.e. one or more fully connected layers.

Looking at the evolution CNNs, the depth appears to be a key feature. On challenging image processing tasks, an increase of performance is often related to a deeper network. As an example, AlexNet [KSH12] has 5 convolutions while VGG16 and VGG19 [SZ14] have respectively 16 and 19 convolutional layers and more recently, in [HSL$^+$16], the authors train a 1200 layer deep network.

Variations in the LeNet structure have also been used to improve convergence. In a Network in Network (NiN) [LCY14], convolutions are mapped with a multilayer perceptron (1x1 convolutions), which prevent overfitting and improved accuracy on datasets such as CIFAR [KH09]. GoogLeNet [SLJ$^+$15] introduced a multiscale approach using the inception module, composed of parallel convolutions with dif-

ferent kernel sizes.

Optimizing such deep architectures can face practical problems such as overfitting or vanishing or exploding gradients. To overcome these issues, several solutions have been proposed such as enhance optimizers [SMDH13], dropout [SHK+14] applying a random reduction of the number of connection in fully connected layers or on convolutional layers [ZK16], intelligent initialization strategies [GB10] or training sub-networks with stochastic depth [HSL+16].

Residual networks [HZRS16a] achieved the state of the art in many recognition tasks including Imagenet [RDS+15] and COCO [LMB+14]. They proved to be easier to optimize. One of the particularities of these networks is to be very deep, up to hundreds of residual layers. More recently, the authors of [ZK16] introduced wide residual networks which reduce the depth compared to usual resnets by using wider convolutional blocks.

To balance the increasing size of CNNs, various work consider reducing the weight number of the network to reduce memory size and or testing speed. We can distinguish three categories. The first kind of approaches consists in statically modifying the architecture to get lighter networks, e.g. the replacement of the fully connected layers by average pooling [SLJ+15, HZRS16a], the replacement and factorization of convolutions [SVI+16] or the weights constrained to be binary [RORF16, CB16]. This work follows this approach as we modify the internal structure of the residual networks to make it lighter. The second category regroups works that dynamically modify the network at training. Among them, [CK14] alternate between regularization and neuron deactivation (forcing weights to zero) to progressively reduce the number of neurons. In Hashnets [CWT+15] uses a hash function to regroup connections that will share the same weights. Finally the third category post process the network to compress it. Some consider weight pruning [SB15, Pra89] and sparsification [LWF+15]. Other try to compress the data via weight matrix factorization [GLYB14, WLW+16, KPY+15]. [HMD15] remove redundant connections and allow weight sharing, As for dynamical network modifications, these compression methods may also apply to our proposed architectures and add a compression step to our optimized ResNet architecture.

3 Sharing Residual Networks

ShaResNets are based on residual networks architectures in which we force the residual blocks in the same stage, i.e. between two spatial dimension reduction, to share the weights of one convolution. In this section, we first present the residual architectures we based our work on, and then, detail the sharing process.

3.1 Residual networks.

The residual networks basic without [HZRS16a] or with pre-activation [HZRS16b] or wide [ZK16] are a sequential stack of residual blocks, with several convolution layers bypassed by parallel branch. The output of block k, $\mathbf{x}_{k+1}$ can be represented as:

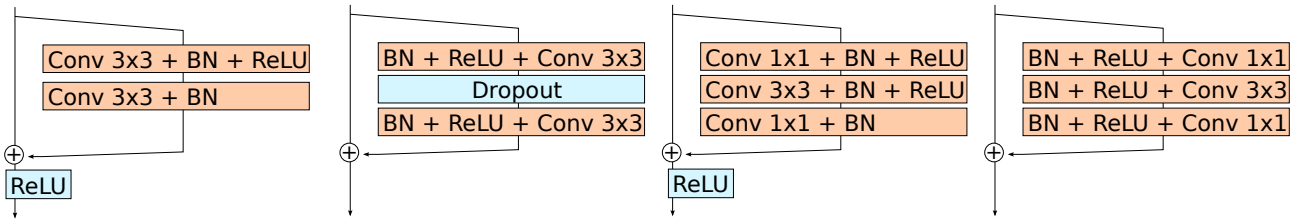
$$\mathbf{x}_{k+1} = \mathbf{x}_k + \mathcal{F}(\mathbf{x}_k, W_k) \tag{1}$$

where $\mathbf{x}_k$ is the input (output of block $k-1$), $\mathcal{F}$ is the residual function and W_k are the parameters of the residual unit. Among them two types of residual blocks are used in this paper.

- *basic* composed of two consecutive 3x3 convolutions.
- *bottleneck* composed of one 3x3 convolution surrounded by two 1x1 convolutions for reducing and then expanding the dimensionality.

A common element of the convolutional structure in the residual blocks is at least a 3x3 convolution. This convolution allows a neighborhood connection so that the final decision is taken using neighborhood relations between pixels and not only independent pixel values.

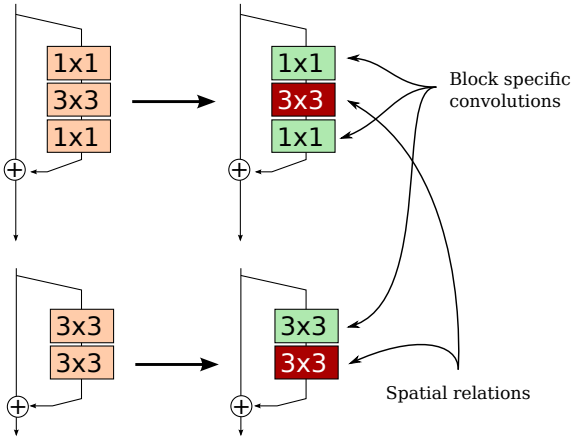
Figure 2 describes the blocks composing the networks presented in this paper. We used two implementations, depending on the datasets (CIFAR 10-100 and ImageNet) to fit the original network structure we will compare to (section 4). They differ in the position of batch normalization and ReLU, before convolutions for CIFAR datasets and after for ImageNet.



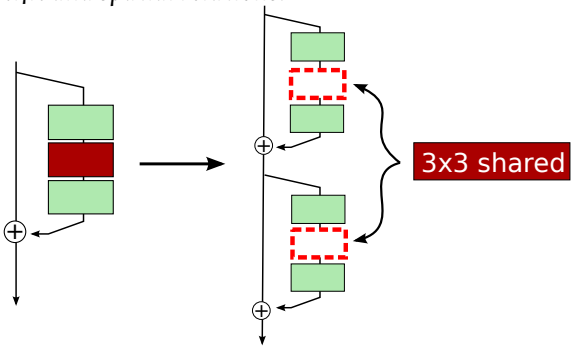
From left to right: Basic Residual Block (ImageNet), Basic Wide Residual Block with optional dropout (CIFAR), Bottleneck Residual Block (ImageNet and CIFAR) and Pre-activation Residual Block (CIFAR).

Figure 2: Residual blocks used for evaluation.

3.2 Sharing weights



(a) *On the left, the original residual block (top: bottleneck, bottom: basic) and on the right, the block discrimination operated before sharing: block specific and spatial relations.*



(b) *The spatial relations are shared between all the residual blocks of the same stage (between two spatial dimension reduction).*

Figure 3: Block specific operations and spatial operations to shared residual block.

The underlying idea of our approach is that it is possible to somehow distinguish two types of mechanisms in a residual block. The first, specific to the block, is the abstraction of the residual block. From block to block, it created higher level features. The second, redundant in the blocks of the same stage, is the spatial connection relations between neighboring tensor cells, between pixels. The equation 1 becomes:

$$\mathbf{x}_{s,k+1} = \mathbf{x}_{s,k} + \mathcal{F}(\mathbf{x}_{s,k}, W_{s,k}, W_s) \qquad (2)$$

where $\mathbf{x}_{s,k}$ (resp. $\mathbf{x}_{s,k+1}$) is the input (resp. the output) of block (k, s), k-th residual unit of stage s. W_k of equation 1 is split into $W_{k,s}$, the parameters specific to the residual block and W_s the parameters shared at the stage level.

In ResNets, the spatial information is taken into account in the 3x3 convolutions and in the layers with dimension reduction (convolution with stride or pooling layers). We first look at the bottleneck block, composed of 3 convolutions (1x1, 3x3 and 1x1). The fictive separation between spatial connection and specific operations is easy as the 1x1 convolution do not connect neighboring cells. This is the top line of figure 3(a). Then, for all spatial connections in the same stage, i.e. for all the blocks between two pooling layers (or convolution with stride), we share the weights. By using a unique 3x3 convolution, we consider that all the spatial connections of a given stage can be explained by a common set of kernels (figure 3(b)).

We adapt the approach to the basic residual block. As it is composed of two 3x3 convolutions, extracting the spatial component is not possible. Still, we adopt a similar approach, the first convolution is considered as specific and the second is shared with the blocks of the same stage (figure 3, bottom line).

In the two cases, we obtain a similar global architecture represented in figure 1. For each stage, the green convolution is common to all block while the

specific items are blue (the number of specific items depends on the architecture choice). The red blocks are the dimensionality reduction layers and the yellow one would be either a multi-layer perceptron or an average pooling.

3.3 Gradient propagation

In modern neural network frameworks, the backward stage is a two step process: gradient computation and actual weight update.In our implementation, the spatial connection layers share weights as well as gradients. At gradient computation, a storage tensor (initialized to zero) is first created for each weight matrix, in our case one tensor for all the convolutional layers sharing the same weights. Then the gradients for each layer are progressively computed from the network output to the input and added to the storage tensor, such that, for the shared convolution at stage s:

$$\nabla_{W_s} = \sum_{i \in S} \nabla_{W_s} J(W_{s,i}) \tag{3}$$

where i stands for the index of the block of stage s and $J(W_{s,k})$ is the objective function.

Finally at weight update step, the weight are modified in the same fashion as with usual non shared convolution according to the optimizer update rule. In our implementation with stochastic gradient descent with momentum the update rule at time t is the same for all layers l:

$$v_{l,t} = \gamma v_{l,t-1} + \alpha \nabla_{W_{l,t}} \tag{4}$$

$$W_l = W_l - v_l \tag{5}$$

where γ is the momentum, α is the learning rate and v is the velocity vector.

4 Experimental results

4.1 Datasets and architectures

We experiment on three dataset, CIFAR 10, CIFAR 100 and ImageNet. We propose evaluations consisting into a comparison between the ShaResNets and their ResNet counterpart.

CIFAR 10 and 100 are two datasets containing 50000 images for training and 10000 for test. In order to show that our sharing process can be generalized to different residual architectures, we use ResNets and Wide ResNets:

- The ResNets implementation (ResNet-164) with 164 convolutions is a good example of very deep network for CIFAR 10 dataset, based on basic residual blocks. (figure 2 first column).

- Wide residual networks are not as deep as the previous but are composed of wider convolutions (more convolutional planes). We present results with two depth: 40 (WRN-40-4) for CIFAR 10 and 28 (WRN-28-10) for CIFAR 100. These are based on the wide residual block (figure 2 second column). The dropout is only activated for CIFAR 100.

Imagenet is a much bigger dataset, with more than one million training images and 1000 classes. We experiment with residual networks of different depth: 34 (ResNet-34) with basic block (figure 2 third column), 50, 101 and 152 (ResNet-50 and ResNet-152) with bottleneck block (figure 2 last column).

4.2 Parameter number reduction

This section deals with the consequences of sharing convolutional weights on the network parameter number. Using one convolution for the spatial relations per stage instead of one per block reduce significantly the size of the network. Comparatively, the deeper the ResNet is the the bigger the gain is for its ShaResNet version.

Table 1 shows the figures for the different architectures and datasets. As expected, the gain is substantial, from 20% for ResNet-50 (ImageNet) to 45% for ResNet-164 (CIFAR). The convolution number in the table expresses the number of independant convolutionnal layers, the shared convolutional layers are counted once.

Dataset	Model	Parameter number		Parameter	Convolution nbr.	
		Orignal	ShaResNet	decrease	Orignal	ShaResNet
CIFAR 10	ResNet-164	1.70 M	0.93 M	45%	164	113
	WRN-40-4	8.95 M	5.85 M	35%	40	25
CIFAR 100	WRN-28-10	36.54 M	26.86 M	26%	28	19
IMAGENET	ResNet-34	21.8 M	13.6 M	37%	34	20
	ResNet-50	25.6 M	20.5 M	20%	50	38
	ResNet-101	44.5 M	29.4 M	33%	101	72
	ResNet-152	60.2 M	36.8 M	39%	152	106

Table 1: Number of parameters of the networks.

4.3 Training

ShaResNet are trained using the same training process as their non-shared counterpart. We used a stochastic gradient descent with momentum and step dropping learning rate policy. CIFAR models were trained with whitened data from PyLearn2 [GWFL$^+$13] and we applied a random horizontal flip on the input image to simulate a larger training dataset and avoid over-fitting. For ImageNet, we adopted a similar approach, we apply on the input image a random crop, random contrast, lighting and color normalization as well as horizontal flip. According to our experiments, the behaviors of our networks are very similar to the original ones. Figure 4 presents the testing accuracy plots obtained on the CIFAR datasets. The gradient accumulation at shared convolutions does not induces instabilities at both training and testing time. The top sub-figure shows the test accuracy of the ResNet-164 architecture on CIFAR-10. Blue curves are the original version [HZRS16a] (third architecture on figure 2) and red the shared version. The plane line is the mean over 5 runs, the colored area represents the standar deviation around the mean. The dot curve curve is the pre-activation version from [HZRS16b] (fourth architecture on figure 2). The two shared architectures perform similarly to their original counter part. The bottom sub-figure presents another experiment on CIFAR-100. We use the Wide ResNet with a widen factor 10 and depth 28, curves are mean over 6 runs. As to the non-shared version, adding a dropout (plain line) increases the performance. However the effect is less significant and while models performs almost the same without dropout, dropout make the original version overcomes the shared one.

4.4 Accuracy

Quantitative evaluation of the networks are presented on table 2. On all these experiments, the top-1 decrease by less than 1% when using the ShaResNet version of the algorithm. The gap between original and shared version is lower on ImageNet and CIFAR 100. This is to be related to the sizes of the networks.

The CIFAR 10 networks are smaller than the others (less than 10M parameters). To our understanding, smaller residual networks induces less redundancy, so that reducing the number of parameters only reduce the learning capacity. On the contrary, large networks such as wide residual network with large widening factor or residual networks for ImageNet are more subject to redundant parameters. In that case, sharing weights makes more sense, like for CIFAR 100 where the accuracy gap is only of 0.2% with parameters reduced by 26%.

Compared to sequential networks, sharing spatial connections at stage level induces a loss of accuracy at test time. We now compare our ShaResNets to less deep networks with a similar number of parameters. Table 3 shows these results for CIFAR. For each shared architecture, we compare its sequential counterpart with reduced depth. We can draw similar conclusion as in the first paragraph. Sharing weights on relatively small architectures (CIFAR 10) is not more efficient than using a less deep network. On the contrary, Wide ResNet with a widen factor of 10 (CIFAR 100) gets a boost in accuracy using shared

Dataset	Model	Error top 1 (%)			Error top 5 (%)		
		Orig.	Share.	Diff.	Orig.	Share.	Diff.
CIFAR 10	ResNet-164	94.54	93.8	0.74			
	WRN-40-4	95.83	94.9	0.93			
CIFAR 100	WRN-28-10 Drop.	80	79.8	0.2			
IMAGENET	ResNet-34	26.73	28.25	0.52	8.74	9.42	0.66
	ResNet-50	24.01	24.61	0.6	7.02	7.41	0.39
	ResNet-101	22.44	22.91	0.47	6.21	6.55	0.34
	ResNet-152	22.16	22.23	0.07	6.16	6.14	−0.02

Table 2: Accuracy on CIFAR and ImageNet. Error percentage (top 1 and top 5) and gap between original and shared version.

convolutions. Deeper networks benefit from mutualisation of spatial relations, the weights are better used, i.e. the ratio accuracy over network size gets better.

Dataset	Model	Param.	Acc.
CIFAR 10	ResNet-164 Share	0.93 M	93.8
	ResNet-92	0.96 M	93.9
	WRN-40-4 Share	5.85 M	94.9
	WRN-28-4	5.85 M	95.0
CIFAR 100	WRN-28-10 Share	26.86 M	79.8
	WRN-22-10	26.85 M	79.55

Table 3: Comparison of accuracies between ShaResNet and ResNets with equivalent size.

On Imagenet, table 2 underlines that the sharing is more efficient as the network goes larger. We even reach similar accuracy (less than 0.2% drop) for the 152 layer architecture. The figure 5 presents the top-1 error function of the weight on ImageNet. The shared architectures plot (green curve) is situated under the blue curve (residual networks). It illustrates that for large networks, shared networks are more efficient than their sequential peers with similar number of parameters. For comparison, we also add the Inception v2 [SVI+16] model performance, in which the authors perform convolution factorization at inception module level to reduce model size.

Timings By reducing the number of parameters in the network and smoothing the gradients (by averaging) at the shared convolution, a gain in training times would be expected. From figure 4, it is obviously not the case, and the same behavior is observed on ImageNet too. Our interpretation is that, at block level, due to the sharing, the weight update may not correspond to the local expected direction of the gradient. In this case, local convolutions (not shared) need to constantly adapt to the new behavior of the shared ones. This latency may explain the similar timings.

4.5 Limitations and perspectives

We have shown in the previous section that residual networks with shared spatial connections are particularly efficient on large networks: given a number of parameters, ShaResNets are more efficient (figure 5). However, they induce a loss in terms of accuracy sometimes even leading to performances similar to networks with reduced depth (CIFAR 10 networks). The conclusions we can draw is that, first, in relatively small networks parameters are used to their potential or at least that redundant spatial connectivities are fewer in number than for large networks.

Second, we chose an arbitrary shared structure. We considered the 3x3 convolutions to operate similar operations for all blocks in the same stage. This assumption may be too restrictive. In our future work we will investigate flexible shared networks, where the sharing rate would be adaptive, function of the noise, the position in the stage and classification dataset (image size, class number). Moreover, we would also investigate other possible splits between spatial relations and block specific operations that would require to modify the basic or bottleneck

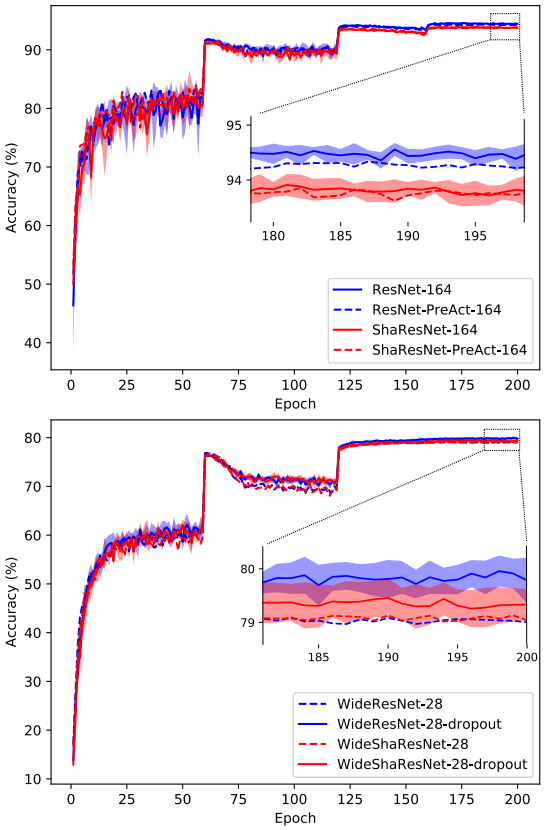
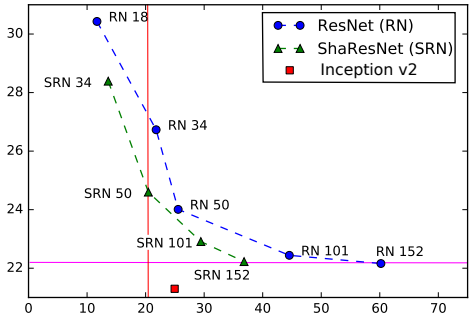


Figure 4: Test accuracies on CIFAR 10 (top) and cifar 100 (bottom).



Red: SRN 50 performs better than RN 34 with less param.
Magenta: SRN 152 and RN 152 have similar performances.

Figure 5: Top 1 error (%) of ShaResNet and ResNets function of the model size (millions of parameters).

residual block structure, for example using channel wise convolution as spatial relations and 1x1 convolution for information abstraction.

Finally, considering only network compression, it will be interesting to test our ShaResNets with compression method mentioned in the related work.

5 Conclusion

In this paper, we introduced the *ShaResNet*, a new convolutional neural network architecture based on residual networks. By sharing convolutions between residual blocks, we create neural architectures lighter than their sequential residual counterpart by 25% to 45% in terms of number of parameters. The training of such network is as easy as with common residual networks. We experimented on three classification datasets. Shared residual architecture proved to be efficient for large networks. We observed an accuracy gap to the corresponding residual architecture of less than 1% for a substantial size reduction. By exploiting the redundant relations of 3x3 convolutions between residual blocks, the ShaResNets make better use of the optimizable weights. With an equivalent parameter number, we obtain better results. We hope that these findings will help further investigation in image processing and more generally in deep learning research.

Implementation details

The experiments uses Torch7. Our code for CIFAR 10 and 100 experiments is based on the original implementations at `github.com/szagoruyko/wide-residual-networks`) and `github.com/facebook/fb.resnet.torch`.
The code is available online at `github.com/aboulch/sharesnet`.

Acknowledgements

This work is part of the DeLTA research project at ONERA `delta-onera.github.io` aiming at exploring machine learning approaches for aerospace applications.

References

[ASL16] Nicolas Audebert, Bertrand Le Saux, and Sébastien Lefevre. Semantic Segmentation of Earth Observation Data Using Multimodal and Multi-scale Deep Networks. *Asian Conference in Computer Vision*, 2016.

[BKC15] Vijay Badrinarayanan, Alex Kendall, and Roberto Cipolla. Segnet: A deep convolutional encoder-decoder architecture for image segmentation. *arXiv preprint arXiv:1511.00561*, 2015.

[BM16] Alexandre Boulch and Renaud Marlet. Deep learning for robust normal estimation in unstructured point clouds. In *Computer Graphics Forum*, volume 35, pages 281–290, 2016.

[BSF94] Yoshua Bengio, Patrice Simard, and Paolo Frasconi. Learning long-term dependencies with gradient descent is difficult, 1994.

[CB16] Matthieu Courbariaux and Yoshua Bengio. Binarynet: Training deep neural networks with weights and activations constrained to +1 or -1. *CoRR*, abs/1602.02830, 2016.

[CK14] Maxwell D Collins and Pushmeet Kohli. Memory bounded deep convolutional networks. *arXiv preprint arXiv:1412.1442*, 2014.

[CMS12] Dan Ciregan, Ueli Meier, and Jürgen Schmidhuber. Multi-column deep neural networks for image classification. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 3642–3649. IEEE, 2012.

[CWT+15] Wenlin Chen, James Wilson, Stephen Tyree, Kilian Weinberger, and Yixin Chen. Compressing neural networks with the hashing trick. In *International Conference on Machine Learning*, pages 2285–2294, 2015.

[GB10] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Aistats*, volume 9, pages 249–256, 2010.

[GDDM14] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Computer Vision and Pattern Recognition*, 2014.

[GLYB14] Yunchao Gong, Liu Liu, Ming Yang, and Lubomir Bourdev. Compressing deep convolutional networks using vector quantization. *arXiv preprint arXiv:1412.6115*, 2014.

[GWFL+13] Ian J. Goodfellow, David Warde-Farley, Pascal Lamblin, Vincent Dumoulin, Mehdi Mirza, Razvan Pascanu, James Bergstra, Frédéric Bastien, and Yoshua Bengio. Pylearn2: a machine learning research library. *arXiv preprint arXiv:1308.4214*, 2013.

[HMD15] Song Han, Huizi Mao, and William J Dally. Deep compression: Compressing deep neural network with pruning, trained quantization and huffman coding. *CoRR, abs/1510.00149*, 2, 2015.

[HS97] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[HSL+16] Gao Huang, Yu Sun, Zhuang Liu, Daniel Sedra, and Kilian Weinberger. Deep networks with stochastic depth. *NIPS 2016, arXiv preprint arXiv:1603.09382*, 2016.

[HZRS16a] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Computer Vision and Pattern Recognition (CVPR), 2016 IEEE Conference on*, 2016.

[HZRS16b] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. *Identity Mappings in Deep Residual Networks*, pages 630–645. Springer International Publishing, 2016.

[JKL+09] Kevin Jarrett, Koray Kavukcuoglu, Yann Lecun, et al. What is the best multi-stage architecture for object recognition? In *2009 IEEE 12th International Conference on Computer Vision*, pages 2146–2153. IEEE, 2009.

[KH09] Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. 2009.

[KPY+15] Yong-Deok Kim, Eunhyeok Park, Sungjoo Yoo, Taelim Choi, Lu Yang, and Dongjun Shin. Compression of deep convolutional neural networks for fast and low power mobile applications. *arXiv preprint arXiv:1511.06530*, 2015.

[KSH12] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[LBBH98] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[LBD+89] Yann LeCun, Bernhard Boser, John S Denker, Donnie Henderson, Richard E Howard, Wayne Hubbard, and Lawrence D Jackel. Backpropagation applied to handwritten zip code recognition. *Neural computation*, 1(4):541–551, 1989.

[LBH15] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521(7553):436–444, 2015.

[LCY14] Min Lin, Qiang Chen, and Shuicheng Yan. Network in network. *ICLR, arXiv preprint arXiv:1312.4400*, 2014.

[LGK16] Isaak Lim, Anne Gehre, and Leif Kobbelt. Identifying Style of 3D Shapes using Deep Metric Learning. *Computer Graphics Forum*, 2016.

[LMB+14] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *European Conference on Computer Vision*, pages 740–755. Springer, 2014.

[LWF+15] Baoyuan Liu, Min Wang, Hassan Foroosh, Marshall Tappen, and Marianna Pensky. Sparse convolutional neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 806–814, 2015.

[Pra89] Lorien Y Pratt. *Comparing biases for minimal network construction with back-propagation*, volume 1. Morgan Kaufmann Pub, 1989.

[RDS+15] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015.

[RHGS15] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster R-CNN: Towards real-time object detection with region proposal networks. In *Advances in Neural Information Processing Systems (NIPS)*, 2015.

[RORF16] Mohammad Rastegari, Vicente Ordonez, Joseph Redmon, and Ali Farhadi. Xnor-net: Imagenet classification using binary convolutional neural networks. *CoRR*, abs/1603.05279, 2016.

[Ros57] Frank Rosenblatt. *The perceptron, a perceiving and recognizing automaton Project Para*. Cornell Aeronautical Laboratory, 1957.

[SB15] Suraj Srinivas and R Venkatesh Babu. Data-free parameter pruning for deep neural networks. *arXiv preprint arXiv:1507.06149*, 2015.

[SHK+14] Nitish Srivastava, Geoffrey E Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

[SL]+15] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–9, 2015.

[SMDH13] Ilya Sutskever, James Martens, George E Dahl, and Geoffrey E Hinton. On the importance of initialization and momentum in deep learning. *ICML (3)*, 28:1139–1147, 2013.

[SVI+16] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jonathon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition,*, 2016.

[SZ14] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[WLW+16] Jiaxiang Wu, Cong Leng, Yuhang Wang, Qinghao Hu, and Jian Cheng. Quantized convolutional neural networks for mobile devices. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4820–4828, 2016.

[ZK16] Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. *BMVC*, 2016.