

Moonshine: Distilling with Cheap Convolutions

Elliot J. Crowley, Gavin Gray, Amos Storkey
 School of Informatics
 The University of Edinburgh

Abstract

Model distillation compresses a trained machine learning model, such as a neural network, into a smaller alternative such that it could be easily deployed in a resource limited setting. Unfortunately, this requires engineering two architectures: a *student* architecture smaller than the first *teacher* architecture but trained to emulate it. In this paper, we present a distillation strategy that produces a student architecture that is a simple transformation of the teacher architecture. Recent model distillation methods allow us to preserve most of the performance of the trained model after replacing convolutional blocks with a cheap alternative. In addition, distillation by attention transfer provides student network performance that is better than training that student architecture directly on data.

1 Introduction

Despite advances in deep learning for a variety of tasks [LeCun et al., 2015], deployment of deep learning into embedded devices e.g. mobile phones, digital cameras, vehicle navigation systems, has been relatively slow due to resource constraints under which these devices operate. Big, memory-intensive neural networks don't fit on these devices, but do these networks have to be big and expensive? The dominant run-time memory cost of neural networks is the number of parameters that need to be stored. Can we have networks with substantially fewer parameters, without the commensurate loss of performance?

It is possible to take a large pre-trained *teacher* network, and use its outputs to aid in the training of a smaller *student* network [Ba and Caruana, 2014] through some distillation process. By doing this the student network is more powerful than if it was trained solely on the training data, and is closer in performance to the larger teacher. The lower-parameter student network typically has an architecture that is more shallow, or thinner – by which we mean its filters have less channels [Romero et al., 2014] – than the teacher. While it is not possible to arbitrarily approximate any network with another [Urban et al., 2017], the limit in neural network performance is at least in part due to the training algorithm, rather than its representational power.

In this paper, we take an alternative approach in designing our student networks. Instead of making networks thinner, or more shallow, we take the standard convolutional block such networks possess and replace it with a *cheaper* convolution block, keeping the original architecture. For example, in a ResNet [He et al., 2016a] this standard block is a pair of sequential 3×3 convolutions. We show that for a comparable number of parameters, student networks that retain the architecture of their teacher but with cheaper convolutional blocks outperform student networks with the original blocks and smaller architectures.

As a model compression strategy, this is very effective. At the same time this transformation

is easy to implement in any deep learning framework; replacing convolutional blocks is a simple substitution into any existing architecture. Furthermore, the optimisation scheme used on the teacher network can be repeated on the student, making another round of hyperparameter optimisation unnecessary.

The cheap convolutional blocks we suggest are described in Section 3 as well as an overview of the methods we employ for distillation. In Section 4 we train a number of student networks for the task of image classification on the CIFAR-10 and CIFAR-100 [Krizhevsky, 2009] datasets and demonstrate that those with cheap convolutions perform better than traditional student networks for a given parameter cost. This level of parameter reduction is competitive with much more complicated methods in the literature [Howard et al., 2017]; and some of these methods may be complementary [Han et al., 2015]. Though it is possible to train the resulting architectures directly, this is demonstrably less effective than distilling them from the larger teacher model.

2 Related Work

The parameters in deep networks have a great deal of redundancy; it has been shown that many of them can be predicted from a subset of parameters [Denil et al., 2013]. However the challenge remains to find good ways to exploit this redundancy without losing model accuracy.

This observation, along with a desire for efficiency improvements has driven the development of smaller, and less computationally-intensive convolutions. One of the most prominent examples is the depthwise separable convolution [Sifre, 2014] which applies a separate convolutional kernel to each channel, followed by a pointwise convolution [Lin et al., 2014] over all channels; depthwise separable convolutions have been used in several architectures [Ioffe and Szegedy, 2015, Chollet, 2016, Xie et al., 2017], and were explicitly adapted to mobile devices in Howard et al. [2017].

However, separating the spatial and channel-wise elements is not the only way to simplify a convolution. In Jin et al. [2015] the authors propose breaking up the general 3D convolution into a set of 3 pointwise convolutions along different axes. The authors of Wang et al. [2016] start with separable convolutions and add topological subdivision, a way to treat sections of tensors separately, and a bottleneck of the spatial dimensions. Both of these methods demonstrate models that are several times smaller than the original model while maintaining accuracy.

In a separable convolution, the most expensive part is the pointwise convolution, so it has been proposed that this operation could also be grouped over sets of channels. However, to maintain some connections between channels, it is helpful to add an operation mixing the channels together [Zhang et al., 2017]. More simply, a squared reduction can be achieved by applying a bottleneck on the channels before the spatial convolution [Xie et al., 2017, Iandola et al., 2016]. In this paper we examine the potency of a separable bottleneck structure.

The work discussed thus far in this section involves learning a compressed network from scratch. There are clear alternatives to this such as retraining after reducing the number of parameters [Han et al., 2015, Li et al., 2017]. We are interested in learning our smaller network as a student through distillation [Ba and Caruana, 2014, Buciluă et al., 2006] in conjunction with a large pre-trained teacher network.

How small can our student be? The complex function of a large, deep teacher network can, theoretically, be approximated by a network with a single hidden layer with enough units [Cybenko, 1989]. The difficulty in practice is *learning that function*. Knowledge distillation [Ba and Caruana, 2014, Hinton et al., 2016] proposes to use the information in the logits of a learnt network to train the smaller student network. In early experiments, this

was shown to be effective; networks much smaller than the original could be trained with small increases in error.

However, modern deep architectures prove harder to compress. For example, a deep convolutional network cannot be trivially replaced by a feedforward architecture [Urban et al., 2017]. Two methods have been proposed to deal with this. First, in Romero et al. [2014] the authors use a linear map between activations at intermediate points to produce an extra loss function. Second, in attention transfer [Zagoruyko and Komodakis, 2017], the authors choose instead to match the activations after taking the mean over the channels. In the context of this paper, we found attention transfer to be effective in our experiments, as described in Section 4.

3 Model Compression with Cheap Convolutions

Given a large, deep network that performs well on a given task, we are interested in compressing that network so that it uses fewer parameters. A flexible and widely applicable way to reduce the number of parameters in a model is to replace all its convolutional layers with a cheaper alternative. Doing this replacement invariably impairs performance when this reduced network is trained directly on the data. Fortunately, we are able to demonstrate that modern distillation methods enable the cheaper model to have performance closer to the original large fully-convolutional network.

3.1 Distillation

For this paper, we utilise and compare two different distillation methods for learning a smaller student network from a large, pre-trained teacher network: knowledge distillation [Hinton et al., 2016, Ba and Caruana, 2014] and attention transfer [Zagoruyko and Komodakis, 2017].

We briefly explain these methods below:

Knowledge Distillation Let us denote the cross entropy of two probability vectors \mathbf{p} and \mathbf{q} as $\mathcal{L}_{CE}(\mathbf{p}, \mathbf{q}) = -\sum_k p_k \log q_k$. Assume we have a dataset of elements, with one such element denoted \mathbf{x} , where each element has a corresponding one-hot class label: denote the one-hot vector corresponding to \mathbf{x} by \mathbf{y} . Given \mathbf{x} , we have a trained teacher network $\mathbf{t} = \text{teacher}(\mathbf{x})$ that outputs the corresponding logits, denoted by \mathbf{t} ; likewise we have a student network that outputs logits $\mathbf{s} = \text{student}(\mathbf{x})$. To perform knowledge distillation we train the student network to minimise the following loss function (averaged across all data items):

$$\mathcal{L}_{KD} = (1 - \alpha)\mathcal{L}_{CE}(\mathbf{y}, \sigma(\mathbf{s})) + 2T^2\alpha\mathcal{L}_{CE}\left(\sigma\left(\frac{\mathbf{t}}{T}\right), \sigma\left(\frac{\mathbf{s}}{T}\right)\right), \quad (1)$$

where $\sigma(\cdot)$ is the softmax function, T is a temperature parameter and α is a parameter controlling the ratio of the two terms. The first term is a standard cross entropy loss penalising the student network for incorrect classifications. The second term is minimised if the student network produces outputs similar to that of the teacher network. The idea being that the outputs of the teacher network contain additional, beneficial information beyond just a class prediction.

Attention Transfer Consider some choice of layers $i = 1, 2, \dots, N_L$ in a teacher network, and the corresponding layers in the student network. At each chosen layer i of the teacher network, collect the spatial map of the activations for channel j into the vector \mathbf{a}_{ij}^t . Let A_i^t collect \mathbf{a}_{ij}^t for all j . Likewise for the student network we correspondingly collect into \mathbf{a}_{ij}^s and A_i^s .

Now given some choice of mapping $\mathbf{f}(A_i)$ that maps each collection of the form A_i into a vector, attention transfer involves learning the student network by minimising:

$$\mathcal{L}_{AT} = \mathcal{L}_{CE}(\mathbf{y}, \sigma(\mathbf{s})) + \beta \sum_{i=1}^{N_L} \left\| \frac{\mathbf{f}(A_i^t)}{\|\mathbf{f}(A_i^t)\|_2} - \frac{\mathbf{f}(A_i^s)}{\|\mathbf{f}(A_i^s)\|_2} \right\|_2, \quad (2)$$

where β is a hyperparameter. Zagoruyko and Komodakis [2017] recommended using $\mathbf{f}(A_i) = (1/N_{A_i}) \sum_{j=1}^{N_{A_i}} \mathbf{a}_{ij}^2$, where N_{A_i} is the number of channels at layer i . In other words, the loss targeted the difference in the spatial map of average squared activations, where each spatial map is normalised by the overall activation norm.

Let us examine the loss (2) further. The first term is again a standard cross entropy loss. The second term, however, ensures the spatial distribution of the student and teacher activations are similar at selected layers in the network, the explanation being that both networks are then *paying attention* to the same things at those layers.

3.2 Cheap Convolutions

As large fully-connected layers are no longer commonplace, convolutions make up almost all of the parameters in modern networks.¹ It is therefore desirable to make them smaller. Here, we present several convolutional blocks that may be introduced in place of a standard block in a network to substantially reduce its parameter cost.

First, let us consider a standard two dimensional convolutional layer that contains N_{out} filters, each of size $N_{\text{in}} \times k \times k$ (assuming square convolutions). N_{out} is the number of channels of the layer output, N_{in} is the number of channels of the input, and $k \times k$ is the kernel size of each convolution. In modern neural networks it is almost always the case that $N_{\text{in}} \leq N_{\text{out}}$. Let $N = \max(N_{\text{in}}, N_{\text{out}})$. Then the parameter cost of this layer is $N_{\text{in}} N_{\text{out}} k^2$, and is bounded by $N^2 k^2$. In a typical residual network, a block contains two such convolutions. We will refer to this as a *Standard* block S , and it is outlined in Table 1.

One alternative to full convolutions, with parameters that scale approximately as N^2 , is to break each convolution into g groups, as shown in Figure 1. By restricting the convolutions to only mix channels within each group and not between groups, we obtain a substantial reduction in the number of parameters for a grouped computation: for example, for $N_{\text{in}} = N_{\text{out}} = N$ the cost changes from $N^2 k^2$ for a standard layer to g groups of $(N/g)^2 k^2$ parameter convolutions, hence reducing the parameter cost by a factor of g . We can then provide some cross-group mixing by following each grouped convolution with a pointwise convolution, with a N^2 parameter cost (when $N_{\text{in}} \neq N_{\text{out}}$ the change in channel size occurs across this pointwise convolution). We refer to this substitution operator as $G(g)$ (grouped convolution with g groups), and illustrate it in Figure 2.

In the original ResNet paper [He et al., 2016a] the authors introduce a bottleneck block which we have parameterised, and denoted as $B(b)$ in Table 1: the input first has its channels decreased by a factor of b via a pointwise convolution, before a full convolution is carried out. Finally, another pointwise convolution brings the representation back up to the desired N_{out} . We can reduce the parameter cost of this block even further by replacing the full convolution with a grouped one; the *Bottleneck Grouped + Pointwise* block is referred to as $BG(b, g)$ and is illustrated in Figure 2.

These substitute blocks are compared in Table 1 and their computational costs (for simplicity we take the case where $N_{\text{in}} = N_{\text{out}} = N$) are given. In practice, by varying the bottleneck size and the number of groups, network parameter numbers may vary over two orders of magnitude; enumerated examples are given in Tables 3 and 4.

¹The parameters introduced by batch normalisation are negligible compared to those in the convolutions. However, they are included for completeness in Table 1.

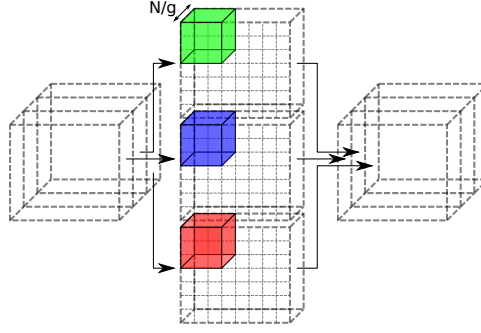


Figure 1: Grouped convolutions operate by passing independent filters over the tensor after it is separated into g groups over the channel dimension. Consider a grouped convolution where the input and output both have N channels: each of the g filters only needs to operate over N/g channels. This reduces the parameter cost of the convolution by a factor of g .

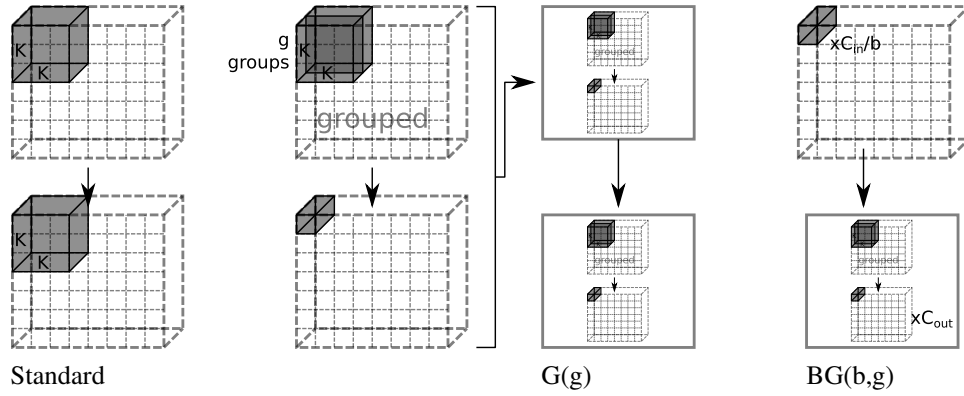


Figure 2: The *Grouped + Pointwise* ($G(g)$) block substitutes all full $k \times k$ convolutions in the *Standard* block with a grouped convolution followed by a pointwise (1×1) convolution. To reduce parameters further, a pointwise *Bottleneck* can be used before the Grouped + Pointwise convolution ($BG(b, g)$).

Using grouped convolutions and bottlenecks are common methods for parameter reduction when designing a network architecture. Both are easy to implement in any deep learning framework. Sparsity inducing methods [Han et al., 2015], or approximate layers [Yang et al., 2015], may also provide advantages, but these are complementary to the approaches here. More structured reductions such as grouped convolutions and bottlenecks can be advantageous over sparsity methods in that the sparsity structure does not need to be stored. The claim of this paper is that structured parameter reductions are sufficient to achieve model compression results in line with state of the art *when using effective model distillation*.

Block	S	$G(g)$	$B(b)$	$BG(b, g)$
Structure	BN+ReLU Conv BN+ReLU Conv	BN+ReLU GConv (g) BN+ReLU Conv1x1 BN+ReLU GConv (g) BN+ReLU Conv1x1	BN+ReLU Conv1x1($N \rightarrow \frac{N}{b}$) BN+ReLU Conv BN+ReLU Conv1x1($\frac{N}{b} \rightarrow N$)	BN+ReLU Conv1x1($N \rightarrow \frac{N}{b}$) BN+ReLU GConv(g) BN+ReLU Conv1x1($\frac{N}{b} \rightarrow N$)
Conv Params	$2N^2k^2$	$2N^2(\frac{k^2}{g} + 1)$	$N^2(\frac{k^2}{b^2} + \frac{2}{b})$	$N^2(\frac{k^2}{gb^2} + \frac{2}{b})$
BN Params	$8N$	$16N$	$N(4 + \frac{8}{b})$	$N(4 + \frac{8}{b})$

Table 1: Convolutional Blocks used in this paper: a standard block S , a grouped + pointwise block G , a bottleneck block B , and a bottleneck grouped + pointwise block BG . All blocks use pre-activations [He et al., 2016b]. Conv refers to a $k \times k$ convolution. GConv is a grouped $k \times k$ convolution and Conv1x1 is a pointwise convolution. BN+ReLU refers to a batch-normal layer followed by a ReLU activation. We assume that the input and output to each block has N channels and that channel size does not change over a particular convolution unless written out explicitly as $(x \rightarrow y)$. Where applicable, g is the number of groups in a grouped convolution and b is the bottleneck contraction. We give the parameter cost of the convolutions in each block in terms of these parameters. The batch-norm parameter cost (assuming running mean/variance are kept for normalisation) is also given, but is markedly smaller.

4 Experiments

In this section we train and evaluate a number of student networks, each distilled from the *same* large teacher network. We distil with (i) knowledge distillation and (ii) attention transfer. We also train the networks without any form of distillation to observe whether the the distillation process is necessary to obtain good performance. In this way we demonstrate that the high performance comes from the distillation, and cannot be achieved by directly training the student networks using the data.

For comparison we also study student networks with smaller architectures (i.e. fewer layers/filters) than the teacher. This enables us to test if the block transformations we propose are key, or it is simply a matter of distilling networks with smaller numbers of parameters. We compare the smaller student architectures with student architectures implementing cheap, substitute convolutional blocks (but with the same architecture as the teacher). The different convolutional blocks are summarised in Table 1 and the student networks are described in detail in Section 4.1.

Experiments are conducted for both the CIFAR-10 and CIFAR-100 datasets. Results for

group	output size	structure
<i>conv1</i>	$16 \times 32 \times 32$	$1 \times \text{Conv3x3}(N = 16)$
<i>conv2</i>	$16k \times 32 \times 32$	$n \times \text{Block}(N = 16k)$
<i>conv3</i>	$32k \times 16 \times 16$	$n \times \text{Block}(N = 32k)$
<i>conv4</i>	$64k \times 8 \times 8$	$n \times \text{Block}(N = 64k)$
<i>pool</i>	$64k \times 1 \times 1$	8×8 avg-pool
<i>fc</i>	<i>classes</i>	$64k \times \text{classes}$ fully connected

Table 2: Summary of the Wide ResNet structures used in experiments; matching those in Zagoruyko and Komodakis [2017]. The bulk of the parameters are in $\{\text{conv2}, \text{conv3}, \text{conv4}\}$ which each consist of n blocks with channel width N controlled by k . We explore the effect of substituting these blocks with cheaper alternatives. *classes* refers to the number of object classes which is, perhaps unsurprisingly, 10 for CIFAR-10 and 100 for CIFAR-100.

CIFAR-10 are given in Table 3 and Figure 3. Results for CIFAR-100 can be found in Table 4 and Figure 4. These results are discussed in detail in Section 4.2.

4.1 Network Descriptions

For our experiments we utilise the competitive Wide Residual Network (WRN) architecture [Zagoruyko and Komodakis, 2016], which is briefly summarised in Table 2. The bulk of the network lies in the $\{\text{conv2}, \text{conv3}, \text{conv4}\}$ groups and the network depth d determines the number of convolutional blocks n in these groups as $n = (d - 4)/6$. The network width, denoted by k , affects the channel size of the filters in these blocks. Note that when we employ attention transfer the student and teacher outputs of groups $\{\text{conv2}, \text{conv3}, \text{conv4}\}$ are used as $\{A_1, A_2, A_3\}$ in the second term of Equation (2) with $N_L = 3$.

For our teacher network we use WRN-40-2 (a WRN with depth 40 and width 2) with standard (S) blocks. 3×3 kernels are used for all non-pointwise convolutions in our student and teacher networks unless stated otherwise. For our student networks we use:

- WRN-40-1, 16-2, and 16-1 with S blocks. These are student networks that are thinner and/or more shallow than the teacher and represent typical student networks most works employ.
- WRN-40-2 with S blocks where the 3×3 kernels have been replaced with 2×2 dilated kernels (as described in [Yu and Koltun, 2016]). This allows us to see if it possible to naively reduce parameters by effectively zeroing out elements of each kernel.
- WRN-40-2 using a bottleneck block B with $2\times$ and $4\times$ channel contraction (b).
- WRN-40-2 using a grouped + pointwise block G for group sizes (g) $\{2, 4, 8, 16, N/16, N/8, N/4, N/2, N\}$ where N is the number of channels in that block. This allows us to explore the spectrum between full convolutions ($g = 1$) and fully separable convolutions ($g = N$).
- WRN-40-2 with a bottlenecked grouped + pointwise block BG . We use $b = 2$ with groups sizes of $\{2, 4, 8, 16, M/16, M/8, M/4, M/2, M\}$ where $M = N/b$ is the number of channels **after the bottleneck**. We use this notation so that ($g = M$) represents fully separable convolutions and we can easily denote divisions thereof. $BG(4, M)$ is also used to observe the effect of extreme compression.

Implementation Details Experiments were conducted in PyTorch [Paszke et al.]. For training we used minibatches of size 128. Before each minibatch, the images were padded

by 4×4 zeros, and then a random 32×32 crop was taken. Each image was left-right flipped with a probability of a half. Training was conducted for 200 epochs using SGD with standard momentum fixed at 0.9 with an initial learning rate of 0.1. The learning rate was reduced by a factor of 0.2 at the start of epochs 60, 120, and 160. For knowledge distillation we set α to 0.9 and used a temperature of 4. For attention transfer β was set to 1000.

4.2 Analysis and Observations

Figure 3 compares the parameter cost of each student network (on a log scale) against the test error on CIFAR-10 obtained with attention transfer. On this plot, the ideal network would lie in the bottom-left corner (few parameters, low error). What is fascinating is that almost every network with the same architecture as the teacher, but with cheap convolutional blocks (those on the blue, green, and cyan lines) performs better for a given parameter budget than the reduced architecture networks with standard blocks (the red line). $BG(2, 2)$ outperforms $16 - 2$ (5.57% vs. 5.66%) despite having considerably fewer parameters (292K vs. 693K). Several of the networks with BG blocks both significantly outperform $16 - 1$ and use less parameters.

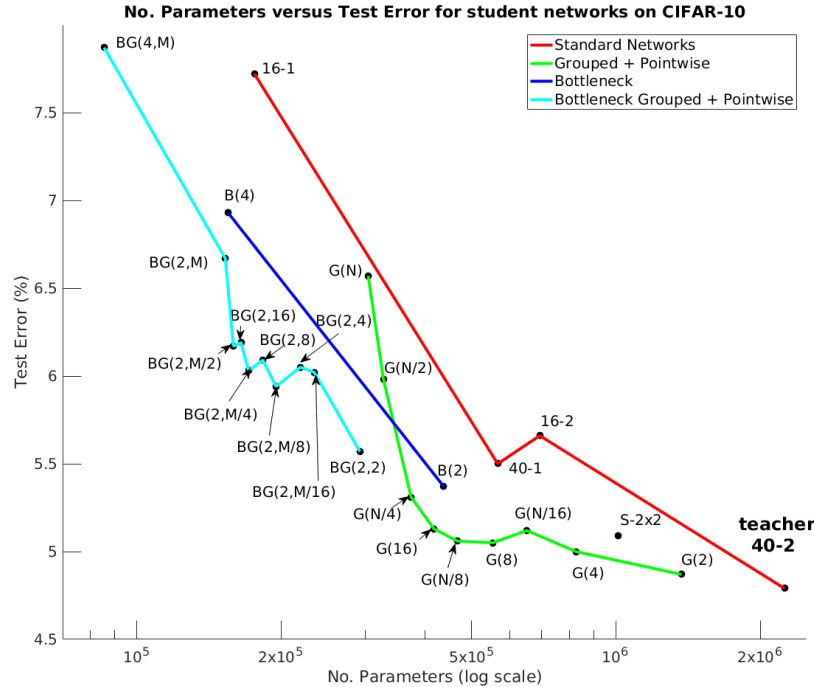


Figure 3: Test Error vs. No. parameters for student networks learnt with attention transfer on CIFAR-10. Note that the x-axis is logarithmically scaled. Points on the red curve correspond to networks with S convolutional blocks and reduced architectures. All other networks have the same WRN-40-2 architecture as the teacher but with cheap convolutional blocks: G (green), B (blue), and BG (cyan). The blocks are described in detail in Table 1. Notice that the student networks with cheap blocks outperform those with smaller architectures and standard convolutions for a given parameter budget.

It is encouraging that significant compression is possible with only small losses; several networks perform almost as well as the teacher with considerably less parameters – $G(N/8)$ (blue) has an error of 5.06%, close to that of the teacher, but has just over a fifth of the parameters. $BG(2, M/8)$ has less than a tenth of the parameters of the teacher, for a cost of 1.15% increase in error. With a similar change in error, these compression rates exceed those found in contemporary papers, such as Cohen and Welling [2016]. Even simply switching all convolutions with smaller, dilated equivalents $S - 2 \times 2$ allows one to use half the parameters for a similar performance.

An important lesson can be learnt regarding grouped + pointwise convolutions. They are often used in their depthwise-separable [Chollet, 2016] form (when the number of groups is equal to N , the total number of channels in the convolution). However, the networks with half, or quarter that number of groups perform substantially better for a modest increase in parameters. $G(N/4)$ has 374K parameters compared to the 304K of $G(N)$ but has an error that is 1.26% lower. As we make the number of groups smaller the performance gets close to that of the teacher as the network structure is getting closer and closer to the original convolutions in the teacher (neglecting the pointwise component). The number of groups is an easy parameter to tune to trade some performance for a smaller network. Grouped + pointwise convolutions also work well in conjunction with a bottleneck of size 2, although for large bottlenecks the error increases rather significantly – as can be seen for $BG(4, M)$. Despite this, it is still of comparable performance to 16 – 1 with half the parameters. We observe similar trends for CIFAR-100 in Figure 4.

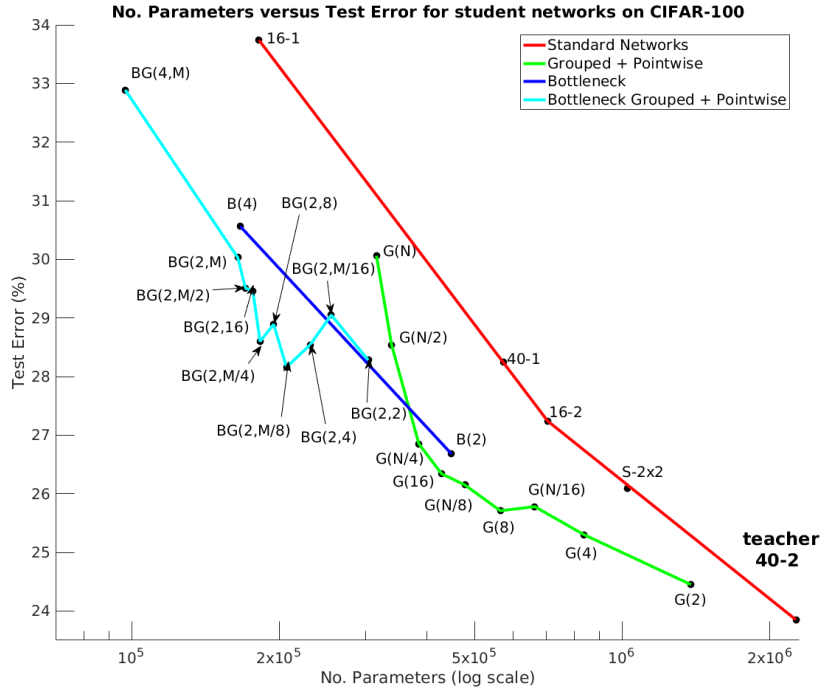


Figure 4: Test Error vs. No. parameters for student networks learnt with attention transfer on CIFAR-100. Points on the red curve correspond to networks with S convolutional blocks and reduced architectures. All other networks have the same WRN-40-2 architecture as the teacher but with cheap convolutional blocks: G (green), B (blue), and BG (cyan).

We also observe that attention transfer from the teacher network is substantially better than knowledge distillation or than training a network structure directly on the data. Consider Table 3, which shows the attention transfer errors of Figure 3 (the AT Error column) alongside those of networks trained with knowledge distillation (KD Error), and no distillation (Error) for CIFAR-10. In all cases, the student network trained with attention transfer is better than the student network trained by itself – a distillation process appears to be necessary. Some performances are particularly impressive – for $G(2)$ blocks the error is only 0.08% higher than the teacher despite the network having under half as many parameters. It is also noticeable that knowledge distillation gives similar, or even worse performance than if the student network was trained by itself. These conclusion are mirrored for CIFAR-100 training (Table 4).

Romero et al. [2014] note that the performance issues with knowledge distillation occur in networks with a depth of more than 7 layers. Zagoruyko and Komodakis [2017] also observe that experiments training a CNN on Imagenet *would not converge* when using knowledge distillation.

5 Conclusion

After training a large, deep model it may be prohibitively time consuming to design a model compression strategy in order to deploy it. On many problems, it may also be more difficult to achieve the desired performance with a smaller model. We have demonstrated a model compression strategy that is fast to apply, and doesn’t require any additional engineering. Furthermore, the optimisation algorithm of the larger model is sufficient to train the cheaper student model.

The cheap convolutions used in this paper were chosen for their ease of implementation. Future work could investigate more complicated approximate operations, such as those described in Moczulski et al. [2015]; which could make a difference for the 1×1 convolutions in the final layers of a network. One could also make use of custom blocks generated through a large scale black box optimisation as in [Zoph et al., 2017]. Equally, there are many methods for low rank approximations that could be applicable [Jaderberg et al., 2014, Garipov et al., 2016, Sainath et al., 2013]. We hope that this work encourages others to consider *cheapening their convolutions* as a compression strategy.

Acknowledgements. This project has received funding from the European Union’s Horizon 2020 research and innovation programme under grant agreement No. 732204 (Bonseyes). This work is supported by the Swiss State Secretariat for Education, Research and Innovation (SERI) under contract number 16.0159. The opinions expressed and arguments employed herein do not necessarily reflect the official views of these funding bodies.

Depth-Width	Block	No. Params	Error	KD Error	AT Error
Teacher 40-2	S	2248954	4.79	–	–
16-2	S	693498	6.53	6.03	5.66
40-1	S	566650	6.48	6.39	5.50
16-1	S	175994	8.81	8.75	7.72
40-2	S-2x2	1012474	5.89	6.03	5.09
40-2	G(2)	1369530	5.30	5.37	4.87
40-2	G(4)	825210	5.50	5.81	5.00
40-2	G(8)	553050	5.92	5.72	5.05
40-2	G(16)	416970	6.65	6.38	5.13
40-2	G(N/16)	651834	5.72	5.72	5.12
40-2	G(N/8)	466362	6.07	5.61	5.06
40-2	G(N/4)	373626	6.93	6.45	5.31
40-2	G(N/2)	327258	7.12	6.83	5.98
40-2	G(N)	304074	8.51	8.01	6.57
40-2	B(2)	437242	6.36	6.28	5.37
40-2	B(4)	155002	7.94	7.83	6.93
40-2	BG(2,2)	292090	6.12	6.25	5.57
40-2	BG(2,4)	219514	6.75	6.75	6.05
40-2	BG(2,8)	183226	6.94	6.98	6.09
40-2	BG(2,16)	165082	6.77	6.97	6.19
40-2	BG(2,M/16)	234704	6.26	6.50	6.02
40-2	BG(2,M/8)	195322	6.75	6.49	5.94
40-2	BG(2,M/4)	171130	7.06	7.15	6.03
40-2	BG(2,M/2)	159034	7.45	7.47	6.17
40-2	BG(2,M)	152986	7.95	7.99	6.67
40-2	BG(4,M)	85450	9.04	8.61	7.87

Table 3: Student Network test error on CIFAR-10. Each network is a Wide ResNet with its depth-width given in the first column, and with its block type in the second. S is a standard convolutional block. $S - 2 \times 2$ is a standard block with dilated 2x2 kernels. $G(g)$ is a grouped + pointwise block with g groups. $B(b)$ is a bottleneck block with contraction b . $BG(b, g)$ is a bottleneck block with contraction b and a grouped convolution with g groups. N refers to the channel width of each block, and M refers to the channel width after the bottleneck where applicable. The total parameter cost of the network is given in the third column. Errors are reported for (i) learning with no distillation (Error), (ii) knowledge distillation with a teacher (KD Error), and attention transfer with a teacher (AT Error). The same teacher is used for training, and is given in the first row. This table shows that (i) through attention transfer it is possible to cut the number of parameters of a network, but retain high performance and (ii) for a similar number of parameters, students with cheap convolutional blocks, outperform those with expensive convolutions and smaller architectures.

Depth-Width	Block	No. Params	Error	KD Error	AT Error
Teacher 40-2	S	2260564	23.85	–	–
16-2	S	705108	27.63	27.97	27.24
40-1	S	572500	29.64	30.21	28.24
16-1	S	181844	34	37.28	33.74
40-2	S-2x2	1024084	27.2	26.98	26.09
40-2	G(2)	1381140	25.94	24.92	24.45
40-2	G(4)	836820	26.20	25.48	25.30
40-2	G(8)	564660	26.49	26.64	25.71
40-2	G(16)	428580	28.85	27.10	26.34
40-2	G(N/16)	663444	27.08	26.11	25.78
40-2	G(N/8)	477972	27.85	27.05	26.15
40-2	G(N/4)	385236	28.91	27.93	26.85
40-2	G(N/2)	338868	30.24	28.89	28.54
40-2	G(N)	315684	31.84	29.99	30.06
40-2	B(2)	448852	28.27	28.08	26.68
40-2	B(4)	166612	31.63	33.63	30.56
40-2	BG(2,2)	303700	28.51	28.82	28.28
40-2	BG(2,4)	231124	29.39	29.25	28.54
40-2	BG(2,8)	194836	30.21	29.34	28.89
40-2	BG(2,16)	176692	30.57	30.54	29.46
40-2	BG(2,M/16)	255316	29.69	28.69	29.05
40-2	BG(2,M/8)	206932	29.09	29.13	28.16
40-2	BG(2,M/4)	182740	30.42	30.28	28.60
40-2	BG(2,M/2)	170644	30.44	30.66	29.51
40-2	BG(2,M)	164596	30.90	31.18	30.03
40-2	BG(4,M)	97060	33.64	37.34	32.89

Table 4: Student Network test error on CIFAR-100. Each network is a Wide ResNet with its depth-width given in the first column, and with its block type in the second. These blocks are described in detail in Section 3.2. The total parameter cost of the network is given in the third column. Errors are reported for (i) learning with no distillation (Error), (ii) knowledge distillation with a teacher (KD Error), and attention transfer with a teacher (AT Error). The same teacher is used for training, and is given in the first row.

References

- Lei Jimmy Ba and Rich Caruana. Do deep nets really need to be deep? In *Advances in Neural Information Processing Systems*, 2014.
- Cristian Buciluă, Rich Caruana, and Alexandru Niculescu-Mizil. Model compression. In *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2006.
- François Chollet. Xception: Deep learning with depthwise separable convolutions. *CoRR*, abs/1610.02357, 2016. URL <http://arxiv.org/abs/1610.02357>.
- Taco Cohen and Max Welling. Group equivariant convolutional networks. In Maria Florina Balcan and Kilian Q. Weinberger, editors, *Proceedings of The 33rd International Conference on Machine Learning*, volume 48 of *Proceedings of Machine Learning Research*, pages 2990–2999, New York, New York, USA, 20–22 Jun 2016. PMLR. URL <http://proceedings.mlr.press/v48/cohen16.html>.
- George Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals, and Systems (MCSS)*, 2(4):303–314, 1989.
- Misha Denil, Babak Shakibi, Laurent Dinh, Marc’Aurelio Ranzato, and Nando de Freitas. Predicting parameters in deep learning. In *Advances in Neural Information Processing Systems*, 2013.
- Timur Garipov, Dmitry Podoprikin, Alexander Novikov, and Dmitry P. Vetrov. Ultimate tensorization: compressing convolutional and FC layers alike. *CoRR*, abs/1611.03214, 2016. URL <http://arxiv.org/abs/1611.03214>.
- Song Han, Huizi Mao, and William J. Dally. Deep compression: Compressing deep neural networks with pruning, trained quantization and Huffman coding. *CoRR*, abs/1510.00149, 2015. URL <http://arxiv.org/abs/1510.00149>.
- Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016a.
- Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Identity mappings in deep residual networks. In *European Conference on Computer Vision*, 2016b.
- Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *CoRR*, abs/1503.02531, 2016. URL <http://arxiv.org/abs/1503.02531>.
- Andrew G. Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. MobileNets: Efficient convolutional neural networks for mobile vision applications. *CoRR*, abs/1704.04861, 2017. URL <http://arxiv.org/abs/1704.04861>.
- Forrest N. Iandola, Matthew W. Moskewicz, Khalid Ashraf, Song Han, William J. Dally, and Kurt Keutzer. SqueezeNet: AlexNet-level accuracy with 50× fewer parameters and < 1MB model size. *CoRR*, abs/1602.07360, 2016. URL <http://arxiv.org/abs/1602.07360>.
- Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International Conference on Machine Learning*, pages 448–456, 2015.
- Max Jaderberg, Andrea Vedaldi, and Andrew Zisserman. Speeding up convolutional neural networks with low rank expansions. In *British Machine Vision Conference*, 2014.
- Jonghoon Jin, Aysegul Dundar, and Eugenio Culurciello. Flattened convolutional neural networks for feedforward acceleration. In *International Conference on Learning Representations*, 2015.

- Alex Krizhevsky. Learning multiple layers of features from tiny images. Master’s thesis, University of Toronto, 2009.
- Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521(7553): 436–444, 2015.
- Zhe Li, Xiaoyu Wang, Xutao Lv, and Tianbao Yang. SEP-Nets: Small and effective pattern networks. *CoRR*, abs/1706.03912, 2017. URL <http://arxiv.org/abs/1706.03912>.
- Min Lin, Qiang Chen, and Shuicheng Yan. Network in network. In *International Conference on Learning Representations*, 2014.
- Marcin Moczulski, Misha Denil, Jeremy Appleyard, and Nando de Freitas. ACDC: a structured efficient linear layer. *CoRR*, abs/1511.05946, 2015. URL <http://arxiv.org/abs/1511.05946>.
- Adam Paszke, Sam Gross, Soumith Chintala, and Gregory Chanan. PyTorch: Tensors and dynamic neural networks in Python with strong GPU acceleration. <https://github.com/pytorch/pytorch>. Accessed: 31st October 2017.
- Adriana Romero, Nicolas Ballas, Samira Ebrahimi Kahou, Antoine Chassang, Carlo Gatta, and Yoshua Bengio. FitNets: Hints for thin deep nets. *CoRR*, abs/1412.6550, 2014. URL <http://arxiv.org/abs/1412.6550>.
- Tara N. Sainath, Brian Kingsbury, Vikas Sindhwani, Ebru Arisoy, and Bhuvana Ramabhadran. Low-rank matrix factorization for deep neural network training with high-dimensional output targets. In *IEEE International Conference on Acoustics, Speech and Signal Processing*, 2013.
- Laurent Sifre. *Rigid-Motion Scattering for Image Classification*. PhD thesis, École Polytechnique, 2014.
- Gregor Urban, Krzysztof J. Geras, Samira Ebrahimi Kahou, Ozlem Aslan, Shengjie Wang, Rich Caruana, Abdelrahman Mohamed, Matthai Philipose, and Matt Richardson. Do deep convolutional nets really need to be deep and convolutional? In *International Conference on Learning Representations*, 2017.
- Min Wang, Baoyuan Liu, and Hassan Foroosh. Factorized convolutional neural networks. *CoRR*, abs/1608.04337, 2016. URL <http://arxiv.org/abs/1608.04337>.
- Saining Xie, Ross Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He. Aggregated residual transformations for deep neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017.
- Zichao Yang, Marcin Moczulski, Misha Denil, Nando de Freitas, Alex Smola, Le Song, and Ziyu Wang. Deep fried convnets. In *Proceedings of the IEEE International Conference on Computer Vision*, 2015.
- Fisher Yu and Vladlen Koltun. Multi-scale context aggregation by dilated convolutions. In *International Conference on Learning Representations*, 2016.
- Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. In *British Machine Vision Conference*, 2016.
- Sergey Zagoruyko and Nikos Komodakis. Paying more attention to attention: Improving the performance of convolutional neural networks via attention transfer. In *International Conference on Learning Representations*, 2017.
- X. Zhang, X. Zhou, M. Lin, and J. Sun. ShuffleNet: An extremely efficient convolutional neural network for mobile devices. *CoRR*, abs/1707.01083, 2017. URL <http://arxiv.org/abs/1707.01083>.
- Barret Zoph, Vijay Vasudevan, Jonathon Shlens, and Quoc V. Le. Learning transferable architectures for scalable image recognition. *CoRR*, abs/1707.07012, 2017. URL <http://arxiv.org/abs/1707.07012>.