DecomposeMe: Simplifying ConvNets for End-to-End Learning

Jose M. Alvarez and Lars Petersson

NICTA / Data61, Canberra, Australia
{jose.alvarez,lars.petersson}@data61.csiro.au

Abstract. Deep learning and convolutional neural networks (ConvNets) have been successfully applied to most relevant tasks in the computer vision community. However, these networks are computationally demanding and not suitable for embedded devices where memory and time consumption are relevant.
In this paper, we propose **DecomposeMe**, a simple but effective technique to learn features using 1D convolutions. The proposed architecture enables both simplicity and filter sharing leading to increased learning capacity. A comprehensive set of large-scale experiments on ImageNet and Places2 demonstrates the ability of our method to improve performance while significantly reducing the number of parameters required. Notably, on Places2, we obtain an improvement in relative top-1 classification accuracy of 7.7% with an architecture that requires 92% fewer parameters compared to VGG-B. The proposed network is also demonstrated to generalize to other tasks by converting existing networks.
Keywords: Convolutional Neural Networks, separable filters.

1 Introduction

Deep Architectures and, in particular, convolutional neural networks (ConvNets) have experienced great success in recent years. However, while being able to successfully tackle a wide range of challenging problems, current architectures are often limited by the need for large amounts of memory and computational capacity.

In this paper, we set out to alleviate these issues where possible and, in particular, we consider ConvNets used for computer vision tasks, as typically the issues of memory and computation become paramount in that context. Networks useful for real-world tasks may sometimes require as much as a few hundred million parameters [1] to produce state-of-the-art results which increases the memory footprint as well as the computational need. Unfortunately, this means that it is hard to deploy applications where memory and computational resources are relevant such as portable devices. In this work, we demonstrate that the use of filter compositions can not only reduce the number of parameters required to train large scale networks, but also provide better classification performance as evidenced by our experimental results.

We identify two bottlenecks in current convolutional neural network models: computation and memory. While the most computationally expensive operations occur in the first few convolutional layers [2], the larger memory footprint is typically caused by the later, fully-connected, layers. Here, we focus on the first bottleneck mentioned and propose a new architecture intended to speed up the first set of convolutional layers

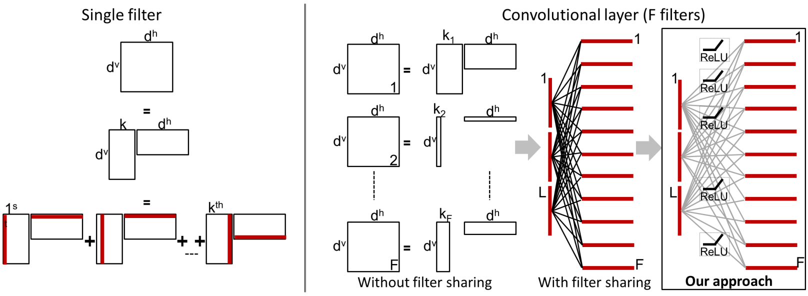


Fig. 1. Left: A 2D tensor (filter) of rank-k can be represented as the product of two matrices or, alternatively, as the linear combination of the outer product of a set of vectors. **Right:** A convolutional layer in a ConvNet consisting of F rank-k filters (not necessarily the same rank or rank-1) can be represented as a linear combination of 1D filters. Furthermore, these filters can be shared within the layer to minimize redundancy. Our approach benefits from this decomposition and, considers compositions of linearly rectified 1D filters.

while maintaining or surpassing the original performance. Further, as a consequence of the improved learning capacity of the network, our approach indirectly alleviates the second bottleneck leading to a significant reduction in the memory footprint.

Many of the current approaches attempting to reduce the computational need have relied on the hope that learned ND filters are low rank enough such that they can be approximated by separable filters [2,3,4]. The main advantages of these approaches are computational cost if the filters are large and reduction in the number of parameters in convolutional layers. However, these methods require a pre-trained network using complete filters and a post processing step to fine-tuning the network and minimize the drop in performance compared to the pre-trained network.

Our approach is different from those mentioned above. We propose **DecomposeMe**, a novel architecture based on 1D convolutions depicted in Figure 1. This architecture introduces three main novelties. *i)* Our architecture relies on imposing separability as a hard constraint by directly learning 1D filter compositions. The fundamental idea behind our method is the fact that any matrix (2D tensor) can be represented as a weighted combination of separable matrices. Therefore, existing architectures can be adequately represented by composing 2D filter kernels by a combination of 1D filters (1D tensors). *ii)* Our proposal further improves the compactness of the model by sharing filters within the convolutional layers. In this way, the proposed network minimizes redundancy and thus further reduces the number of parameters. *iii)* Our proposal improves the learning capacity of the model by inserting a non-linearity in between the 1D filter components. With this modification, the effective depth of the network increases which is intimately related to the number of linear regions available to approximate the sought after function [5]. As a result, we obtain compact models that do not require a pre-trained network and minimize the computational cost and the memory footprint compared to their equivalent networks using 2D filters. Reduced memory footprint has the additional advantage of enabling larger batch sizes at train time and, therefore, computing better gradient approximations leading to, as demonstrated in our experiments, improved classification performance.

A comprehensive set of experiments on four datasets including two large-scale datasets such as Places2 and ImageNet shows the capabilities of our proposal. For instance, on Places 2, compared to a VGG-B model, we obtain a relative improvement in top-1 classification accuracy of 7.7% using 92% fewer parameters compared to the baseline and with a speed up factor of $4.3x$ in a forward-backward pass. Additional experiments on stereo matching also demonstrate the general applicability of the proposed architecture.

2 Related work

In the last few years, the computer vision community has experienced the great success of deep learning. The performance of these end-to-end architectures has continuously increased and outperformed traditional hand-crafted systems. An essential component to their success has been the increment of data available as well as the availability of more powerful computers making possible the training of larger and more computationally demanding networks. For instance, in 2012 the AlexNet [6] model was proposed and won the ImageNet classification challenge with a network that had approximately 2.0M parameters in the convolutional layers (i.e., excluding fully connected ones). More recently, different variations of VGG models were introduced [7] of which VGG-16 has over 14.5M feature parameters. VGG-16 increases the depth of the model by substituting each convolutional kernel with consecutive convolutions consisting of smaller kernels while maintaining the number of filters. As an example, the VGG models as in [7] substitutes 7×7 kernels with 3 consecutive rectified layers of 3×3 kernels. This operation reduces the degrees of freedom compared to the original kernels but at the same time inserts a non-linearity in-between the smaller 3×3 kernels increasing the capacity of the model for partitioning the space [5]. Despite improving the classification performance, the large number of parameters not only makes the training process slow but also makes it difficult to use these models in portable devices where memory and computational resources are relevant.

The growing number of applications deployed in portable devices has motivated recent efforts in speeding up deep models by reducing their complexity. A forerunner work on reducing the complexity of a neural network is the so-called network distillation method proposed in [8]. The idea behind this approach is to train a large, capable, but slow network and then refine this by taking the output of that to train a smaller one. The main strength comes from using the vast network to take care of the regularization process facilitating subsequent training operations. However, this method requires a large pre-trained network to begin with which is not always feasible especially in new problem domains.

Memory-wise the largest contribution comes from the fully connected layers while time-wise the bottleneck is in the first convolutional layers due to a large number of multiplications (larger kernels). In this work, we address the former by simplifying the first convolutional layers. There have been several attempts to reduce the computational cost of these first layers, for example, Denil et. al. [9] proposed to learn only 5% of the parameters and predict the rest based on dictionaries. The existence of this redundancy in the network has motivated other researchers to explore linear structures within the convolutional layers [2,10,11] usually focusing on finding approximations to filters (low-rank filters) by adding constraints in a post-learning process. More specifically, these approaches often learn the unconstrained filter and then approximate the output using a low-rank constraint. For instance, [2] and [10] focus on improving test time by representing convolutional layers as linear combinations of a certain basis. As a result, at test time, a lower number of convolutions is needed to achieve some speeds ups with virtually no drop in performance. Liu et al [11] instead consider sparse representations of the basis rather than linear combinations. However, similar to the distillation process, the starting point of these methods is a pre-trained model.

Directly related to our proposed method is [12] although it is not a convolutional neural network. In that paper, the authors aim at learning separable filters for image processing. To this end, they propose learning a filter combination reinforcing filter separability using low-rank constraints in the cost function. Their results are promising and demonstrate the benefits of learning combinations of separable filters. In contrast to that work, we work within the convolutional layers of a neural network and our filter sharing strategy is different. More importantly, we do not use soft constraints during the optimization process. Instead, we directly enforce filters to be 1D.

3 Simplifying ConvNets through Filter Compositions

In this section we present our DecomposeMe architecture. The essence of our proposal consists of decomposing the ND kernels of a traditional network into N consecutive layers of 1D kernels, see Figure 1. We consider each ND filter as a linear combination of other filters. In contrast to [12] where they seek to find these other filters by solving an optimization problem with additional low-rank constraints, we impose the filters to be 1D and learn them directly from the data. Performance-wise, it turns out that such a decomposition not only mimics the behavior of the original, more complex, network but often surpasses it while being significantly more compact and experiencing a lower computational cost.

For the purpose of clarity, we will here consider 2D filters, however, the analysis is similarly applicable to the ND case. With that in mind, a typical convolutional layer can be analyzed as follows. Let $\mathbf{W} \in \mathbb{R}^{C \times d^h \times d^v \times F}$ denote the weights of a 2D convolutional layer where C is the number of input planes, F is the number of output planes (target number of feature maps) and $d^v \times d^h$ represent the kernel size of each feature map (usually $d^h = d^v \equiv d$). Let $b \in \mathbb{R}^F$ be the vector representing the bias term for each filter. Further, let us now denote $\mathbf{f}^i \in \mathbb{R}^{d^v \times d^h}$ as the ith kernel in the layer. Common approaches first learn these filters from data and then find low-rank approximations as a postprocessing step [10]. However, learned filters may not be separable e.g., specially those in the first convolutional layer [2,10], and these algorithms require an additional fine tuning step to compensate drops in performance.

Instead, it is possible to relax the rank-1 constraint and essentially rewrite $\mathbf{f}^i$ as a linear combination of 1D filters [12]:

$$\mathbf{f}^i = \sum_{k=1}^{K} \sigma_k^i \bar{v}_k^i (\bar{h}_k^i)^T \tag{1}$$

where $\bar{v}_k^i$ and $(\bar{h}_k^i)^T$ are vectors of length d, σ_k^i is a scalar weight, and K is the rank of $\mathbf{f}^i$.

Based on this representation we propose DecomposeMe which is an architecture consisting of decomposed layers. Each decomposed layer represents a N-D convolutional layer as a composition of 1D filters and, in addition, by including a non-linearity $\varphi(\cdot)$ in-between (Figure 1). The i-th output of a decomposed layer, a_i^1, as a function of its input, a_*^0, can be expressed as:

$$a_i^1 = \varphi(b_i^h + \sum_{l=1}^{L} \bar{h}_{il}^T * [\varphi(b_l^v + \sum_{c=1}^{C} \bar{v}_{lc} * a_c^0)]) \tag{2}$$

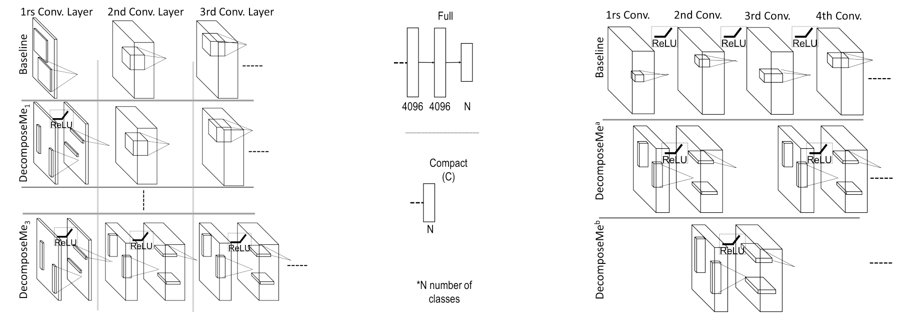
where L represents the number of filters in the intermediate layer. $\varphi(\cdot)$ is set to rectified linear unit (ReLU [6]) in our experiments.

Decomposed layers have two major properties, intrinsically low computational cost, and simplicity. **Computational cost:** Decomposed layers are represented with a reduced number of parameters compared to their original counterparts. This is an immediate consequence of two important concepts: the direct use of 1D filters and the sharing scheme across a convolutional layer leading to greater computational cost savings, especially for large kernel sizes. **Simplicity:** Decomposed architectures are deeper but simpler structures. Decomposed layers are based on filter compositions and therefore lead to smoother (simpler) 2D equivalent filters that help during training by acting as a regularizing factor [2]. Moreover, decomposed layers include a non-linearity in-between convolutions increasing the effective depth of the model. As a direct consequence, the upper bound of the number of linear regions available is increased [13]. Evident from our results, decomposed layers learn faster, as in per epoch, than equivalent 2D convolutional layers. This suggests that the simplicity of the decomposed layers not only reduces the number of parameters but also benefits the training process.

Converting existing structures to decomposed ones is a straight forward process as each existing ND convolutional layer can systematically be decomposed into sets of consecutive layers consisting of 1D linearly rectified kernels and 1D transposed kernels as shown in Figure 1. In the next section we apply decompositions to two well-known computer vision problems such as image classification and stereo matching.

3.1 Complexity Analysis

We analyze the theoretical speed up of the proposed method as follows: Consider as the baseline a convolutional layer of dimensions $C \times F$ with filters of spatial size of $d^v \times d^h$. Without loss of generality, we can assume $d^h = d^v = d$. This baseline is then decomposed into two consecutive layers $C \times L$ and $L \times F$ with filter size $d \times 1$ and $1 \times d$ respectively. The computational cost of these two schemes is proportional to CFd^2 and $L(C+F)d$ respectively. Therefore, considerable improvements are achieved when $L(C+F) << CFd$. The analysis of this expression reveals that, although, d is larger in the first layer (e.g., 11 for AlexNet [14]), C is usually too small compared to L to make a significant difference (e.g., 3 for RGB images). Current architectures tend to have a large number of filters in later layers. For instance, consider a VGG model using kernels of size 3×3, consecutive layers of equal size (e.g., 256), and maintaining the number of output filters through the decomposed layer ($L = 256$). In that case, the theoretical improvement of our method is given by $256(256 + 256)$ vs. $256 \times 256 \times 3$.



(a) (b) (c)

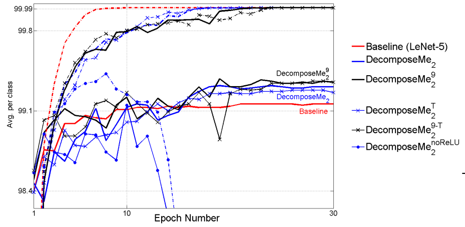
Fig. 2. DecomposeME: a) Convolutional layers in a ConvNet can be represented using our proposed DecomposeMe$_i$ architecture, where i is the number of layers being decomposed. b) We evaluate the proposal using Full and Compact models. The former is the original architecture consisting of two fully connected layers and the final classification layer with 4096,4096,N neurons respectively; The latter, a Compact model, directly connects the output of the last convolutional layer to a layer with N neurons. N is the number of classes. c) Consecutive convolutional layers can also be converted into decomposed ones maintaining the size of the receptive field in the input feature map.

4 Experiments

We conduct two sets of experiments representing different use cases to validate our proposal. Firstly, we run experiments performing image classification. More specifically, we test four well-known network architectures, namely LeNet [15], CIFAR-10 quick [16] and AlexNet [6] and VGG [7], on three publicly available datasets; MNIST [17], CIFAR-10 [18] and ImageNet [19]. An additional experiment is included on the challenging Places2 dataset [20]. Secondly, we run experiments performing stereo matching to show the generic learning capabilities and applicability of our proposal. To this end, we consider a state-of-the-art stereo matching problem and replace the existing, accurate, network of [21] with our decomposed architecture. This set of experiments is carried out on the KITTI benchmark [22].

4.1 Image Classification

All the experiments on image classification are conducted on a Dual Xeon 8-core E5-2650 with 128GB of RAM using two Kepler Tesla K20 GPUs in parallel, unless otherwise specified. We use the torch-7 framework [23] and large-scale experiments are carried out using the multi-GPU implementation available in [24]. Learning rate, weight decay and momentum were set to the default values. More precisely, we start with a learning rate of 0.01 which is decreased when the training error plateaus; weight decay is set to 0.0001 and momentum to 0.9. Again, unless otherwise specified, we use the same hyper-parameter setup as in the original experiments. Data augmentation is done through random crops where necessary and random horizontal flips with probability 0.5. Please note that other training approaches may use different data augmentation techniques such as color augmentation [6]. For a fair comparison, we select the original



(a)

		Avg. (%)	#Params[a]	max d[b]
	Baseline (retrain)	99.1	52.0K	5
LeNet	DecomposeMe$_1^{tan}$	99.2	53.8K	5
	DecomposeMe$_2^{tan}$	99.2	25.6K	5
	DecomposeMe$_2$	99.3	22.0K	5
	LeNet^9	99.2	53.8K	9
	DecomposeMe$_2^9$	99.4	22.0K	9

[a] Total number of parameters.

[b] max d is the largest kernel size in the network.

(b)

Fig. 3. MNIST. (a) Training curves for different instances of our proposal and the baseline. Bold lines represent test accuracy, and dashed ones training accuracy. $noReLU$ stands for layers without non-linearity in-between convolutions. b) Summary of average per class accuracy and number of parameters for different instances of our architecture together with the baseline.

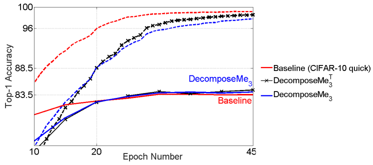
networks as baselines and all models including baselines are trained from scratch on the same computer using the same seed and the same framework.

A basic decomposed layer consists of vertical kernels followed by horizontal ones, and non-linearities in-between 1D convolutions are set to rectifier linear units (ReLU). We evaluate different instances of this model referred to as DecomposeMe$_i^k$ where the sub-index is the number of layers being decomposed (Figure 2a), and the super-index indicates variations in the composition of the layer such as kernel size, the non-linearity being used or the order of the kernels. Decompositions respect the size of the filter in the original model, and the number of output filters from the convolutional layer is maintained. Layers that are not decomposed are left as in the original model. For specific experiments we show results for variations within each of these instances.

MNIST and CIFAR-10 As a sanity check, we first run experiments on the MNIST and CIFAR-10 datasets.

MNIST [17] is a database of handwritten digits, consisting of a training set of 60.000 images and a test set of 10.000 images. All digits in the database have been size-normalized and centered in a fixed-size image. For this experiment we consider the LeNet model proposed in [15] consisting of two convolutional layers with 5×5 kernels, each one followed by max-pooling layers and hyperbolic tangents as non-linear layers, and two fully connected layers. We first gradually substitute convolutional layers for decomposed layers maintaining the number of output filters (referred to as DecomposeMe$_i^{tan}$ since this model keeps the hyperbolic tangent between convolutional layers). Then, we conduct an additional experiment setting the non-linearities between the convolutional layers to rectified linear units. In this case, we also consider a larger kernel size of 9 referred to as DecomposeMe$_2^9$.

Figure 3 summarizes the results for the baseline together with four instances of our proposal. As shown, decompositions systematically outperform the baseline and, when multiple layers are decomposed, significantly reduce the number of parameters in the network. In addition, the performance improves for larger kernel sizes in the first layer. Performance curves for these and additional instances with different filter compositions or excluding the non-linearity in-between decomposed layers are shown in Figure 3a. As shown by DecomposeMe$_i^{noReLU}$, adding the non-linearity is necessary. Looking



(a)

		Top-1	#Params	max d
CIFAR-10 quick	Baseline (retrained)	83.8	5M	5
	DecomposeMe$_1$	84.2	4.7M	5
	DecomposeMe$_3^s$	84.4	3.6M	5
	DecomposeMe$_3$	83.5	2.4M	5
	DecomposeMe$_3^2$	81.2	800K	5

(b)

Fig. 4. CIFAR-10. a) Training curves for different instances of our proposal and the baseline. As in Figure 3, bold lines represent test accuracy, dashed ones training accuracy and a marker indicates a variation in the composition of the layer. b) Top-1 accuracy and summary of the architecture for different instances of our proposal together with the baseline. The super-index refers to variations of the parameter L, see Eq. (2).

at the graph one can see that the structure without non-linearity learns adequately at the beginning and then performance drops drastically after a few iterations. Large scale experiments presented in the next section will also confirm the need of a non-linearity in between 1D convolutions. More importantly, as a consequence of the reduced number of parameters, the gap between training and testing accuracy decreases when using decomposed layers which indicates that the structures are less prone to overfitting. This is evident for instance in the 10th epoch where our proposed method provides similar test accuracy to the baseline, however, the training data accuracy is significantly larger for the baseline, see Figure 3.

CIFAR-10 [18] is a database consisting of 50.000 training and 10.000 testing RGB images with a resolution of 32×32 pixels split into 10 classes. We consider the CIFAR-10 quick model consisting of 5 convolutional layers with kernels of size 5×5 [16].

Figure 4 summarizes the results for the baseline together with an instance of our structure decomposing one layer followed by three instances decomposing all convolutional layers with different L in-between 1D convolutions. As shown, decomposing a single layer provides a slight increment in performance while maintaining the number of parameters. Decomposing additional layers reduces the number of parameters considerably while there is only a slight drop in performance when the reduction is over 50%. We have also experimented with different configurations regarding the decomposition –such as horizontal kernels followed by vertical ones and vice versa, or the combination of both– to verify that the learning process is able to deal with different types of signals. Figure 4a shows learning curves for the baseline versus our structure with three decomposed layers varying the filter composition: vertical convolution followed by a horizontal one and vice versa (referred to as Decomposed$_3^T$). As we can see in these plots, decomposed layers provide a smaller gap between training and testing accuracy and thus reduce overfitting while maintaining performance. For instance, after 20 epochs, all the structures provide the same test performance but the training performance of the baseline is 8% higher. These and additional empirical results (not reported) show that the performance is invariant to permutations of the order of the tensors and that there is no significant benefit in combining two types of configurations. Similarly, we have also experimented with substituting the basic architecture for a Network in Network [25] implementation which renders similar benefits when only the first layer is decomposed. In that case, our architecture achieves an increment of 1% in performance.

Table 1. ImageNet-Places2: Summary of top-1 accuracy on the validation set for instances of our proposal and baselines on (a) ImageNet and (b) Places2. ConvP stands for the number of parameters in 2D convolutional layers. FCP stands for the number of parameters in fully connected layers and $\max d$ refers to the largest kernel size in the network.

			Top-1	#ConvP	#FCP
AlexNetOWTBn	Full	MatConvNet [30]	57.9	2.47M	58.6M
		Baseline (retrain)	57.1	2.47M	58.6M
		DecomposeMe$_1$	59.0	2.47M	58.6M
		DecomposeMe$_1^{xl}$	59.1	2.47M	58.6M
		DecomposeMe$_1^T$	61.1	2.30M	58.6M
		DecomposeMe$_2$	61.3	2.32M	58.6M
		DecomposeMe$_3$	61.8	2.10M	58.6M
		DecomposeMe$_3^{xl}$	59.4	933K	58.6M
	Compact	AlexNetOWTBn^C	54.7	2.47M	9.2M
		DecomposeMe$_3^C$	61.3	2.10M	9.2M
		DecomposeMe$_4^C$	57.8	1.12M	9.2M
B-net	Full	Baseline (retrain)	62.5	9.4M	123.5M
		DecomposeMe$_5^X$	57.5	2.4M	123.5M
		DecomposeMe$_5$	57.8	2.4M	123.5M
	Compact	B-Net^C	61.1	9.4M	25.0M
		DecomposeMe$_5^{C\text{-}X}$	56.9	2.4M	25.0M
		DecomposeMe$_5^C$	57.0	2.4M	25.0M
		DecomposeMe$_8^C$	65.4	7.0M	8.19M
		DecomposeMe$_8^{C-\mathrm{avg}}$	66.2	7.0M	512K

(a)

			Top-1	#ConvP	#FCP
AlexNetOWTBn	Full	Baseline (retrain)	44.5	2.47M	56.1M
		DecomposeMe$_5$	45.2	1.52M	56.1M
	Compact	AlexNetOWTBn^C	41.1	2.47M	3.7M
		DecomposeMe$_3^C$	43.5	2.10M	3.7M
B-net	Full	Baseline (retrain)	44.0	9.4M	121M
		DecomposeMe$_6$	43.8	3.0M	121M
		DecomposeMe$_6^B$	43.6	4.3M	121M
	Compact	B-Net^C	43.1	9.4M	10M
		DecomposeMe$_6^C$	43.8	3.1M	10M
		DecomposeMe$_5^C$	41.3	2.7M	10M
		DecomposeMe$_8^{C-256}$	47.4	7.0M	3.2M

(b)

Large-Scale Experiments: ImageNet and Places2

Datasets. We now focus on two large-scale datasets: ImageNet [26] and Places2 [20]. ImageNet is a large-scale dataset with over 15 million labelled images split into 22.000 categories. We used the ILSVRC-2012 [19] subset of images consisting of 1.2 million images for training and 50.000 images for validation. Places2 [20] is a large-scale dataset created specifically for training systems targeting high-level visual understanding [20] tasks. This dataset consists of more than 10 million training images with 401 unique scene categories and 20000 images for validation. The database comprises between 5000 and 30000 training images per category which is consistent with real-world frequencies of occurrence.

Deep Models. We consider two network structures: the AlexNetOWTBn in [14] and the B-net in [7](VGG-B). AlexNetOWTBn is the "one weird trick" variation (OWT) of AlexNet [6] where we adopt batch normalization (Bn) after each convolutional layer [27]. B-net [7] is the B version of the VGG structure and consists of 10 convolutional layers with max-pooling every two of these convolutions. We consider decompositions in each of those layers, reducing the number of kernels where appropriate. For B-net models, we consider two types of weight initialization: Xavier [28] (referred to as DecomposeMe$_i^X$) and Kaiming [29] which we adopt as default configuration since we obtained slightly better results in this case. In both cases, bias terms were set to 0. Models were trained for a total of 55 epochs with 10000 batches per epoch and a batch size of 96 and 24 for AlexNetOWTBn and B-net respectively.

Network Analysis. We analyze several modifications of the models to better understand the contribution of the proposed approach. First, we study the effect of including

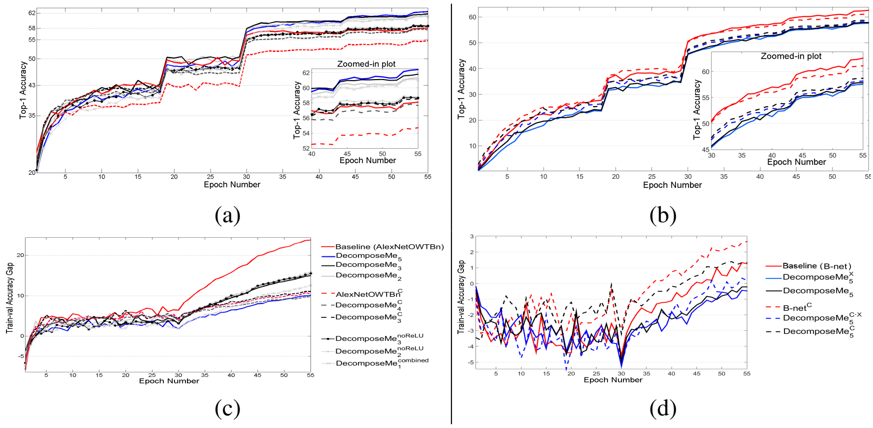
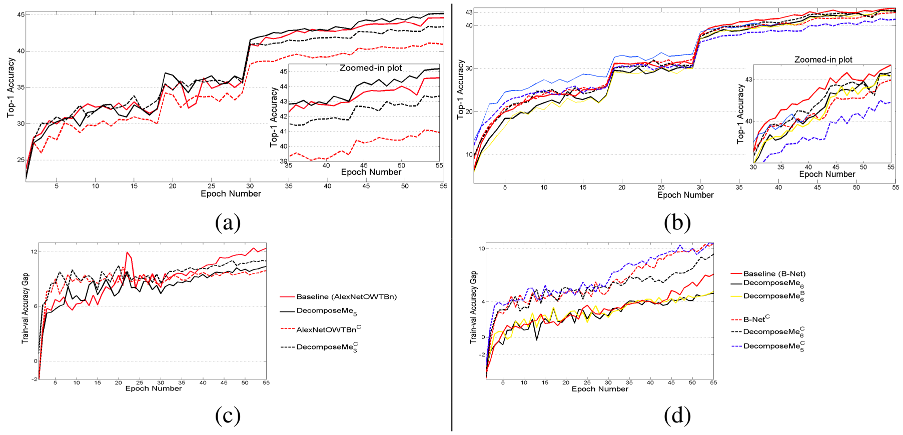


Fig. 5. ImageNet. Training curves for representative instances of our proposal and the baselines. (a) and (c) show top-1 accuracy on the validation set and train-val gap plots respectively for instances of AlexNetOWTBn. (b) and (d) show the corresponding curves for instances of B-net.

non-linearities in-between convolutional layers and different types of filter compositions such as horizontal kernels followed by vertical ones and vice versa, or the combination of both. Second, following the trend of recent architectures [31,32,33] we remove intermediate fully connected layers of the models to compare the performance of the convolutional layers. These compact models solely include a fully connected layer to produce the desired number of outputs (1000 and 401 neurons in ImageNet and Places2 respectively). Figure 2b shows a comparison between original and compact models. As a direct consequence of removing fully connected layers, the number of parameters drops drastically. For comprehensive comparison we also train and report results for the baselines models in their compact form. Compact models do not use DropOut [34].

Evaluation. We measure classification performance as the top-1 accuracy on the validation set using the center crop, named **Top-1**. We also provide training-validation accuracy gap plots, named **Train-val gap**. This plot demonstrates the evolution of the difference between train and validation accuracy as the training proceeds [35]. Overfit models tend to produce a high (positive) gap while underfit models tend to have a similar performance and, therefore, produce a low train validation accuracy gap.

Experimental results. A summary of the results is listed in Table 1a and Table 1b for ImageNet and Places2 respectively. Training plots for selected instances of AlexNetOWTBn and B-Net are shown in Figure 5 and in Figure 6 for ImageNet and Places2 respectively. As shown in Table 1a, for AlexNetOWTBn, the number of parameters is only reduced when more than one layer is decomposed. This is expected since, in the AlexNetOWTBn structure, each decomposed layer introduces an additional convolutional layer (see the complexity analysis in Sect. 4.2). However, despite the slightly larger number of parameters, there is an increment in performance. Empirically, we find similar results when a single layer on OverFeat [1] is decomposed. In that case, there is a performance increment of 2% with respect to the baseline. This suggests that simplified kernels (compositions of 1D kernels) actually help during the training process and the effective capacity of the models increases with the additional non-linear layers.



(a) (b)

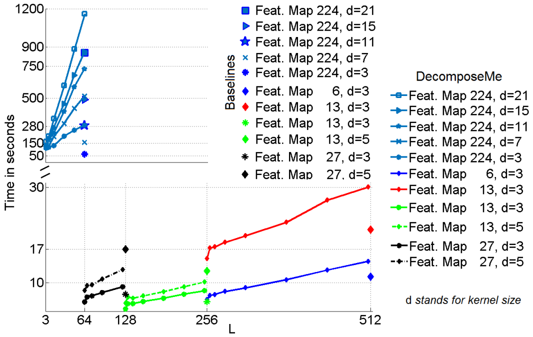
(c) (d)

Fig. 6. Places2. Training curves for representative instances of our proposal and the baselines. (a) and (c) show top-1 accuracy on the validation set and train-val gap plots respectively for instances of AlexNetOWTBn. (b) and (d) show the corresponding curves for instances of B-net.

More substantial changes occur when additional layers are decomposed. The network is then able to produce better results and at the same time reduce the amount of parameters being used. The reduction in the number of parameters is even more substantial when the third layer is decomposed. In this case, the model is still able to perform better than the baseline using only 37.5% of the parameters with respect to the baseline. These results suggest that simplifying ConvNets using the proposed decomposition method not only reduces the amount of parameters required but also outperforms equivalent models learning the complete filter. As in the MNIST experiments, we see no significant difference between variations in the composition of the filters such as horizontal kernels followed by vertical ones (referred to as DecomposeMe$_i^T$). Therefore, we select vertical kernels followed by horizontal ones as the default choice which leads to computational benefits due to memory alignment.

Training curves comparing the effect of including non-linearities in-between decomposed layers are shown in Figure 5a (referred to as DecomposeMe$_2^{noReLU}$ and DecomposeMe$_3^{noReLU}$). As shown, models including non-linearities outperform their equivalent not using rectified kernels independently of the number of decomposed layers. These results suggests that the additional non-linearity in-between each decomposed layer increases the effective capacity of the structure. Interestingly, we can also see in sub figures c and d of Figure 5 and Figure 6 that decomposed layers consistently produce training curves with a smaller gap between training and validation accuracy. From these results, we can infer that low-rank filters help in the regularization process during training. These results are in line with the conclusions drawn in [36]. From these results, we can conclude that our proposed method is less prone to overfitting measured as the gap between training and validation accuracy.

We now focus on results obtained using compact networks (referred to using ^{C}). First, in Figure 5a and Figure 6a we can see that compact instances of AlexNetOWTBn using decomposed layers outperform their equivalent using fully connected layers. For



(a)

	Model	Forward Time	Total Time
AlexNetOWTBn	AlexNetOWTBn [37]	22.45	69.04
	AlexNetOWTBn^C	19.98	58.11
	DecomposeMe$_3$	28.90	90.17
	DecomposeMe$_2$	28.38	88.59
	DecomposeMe$_3^C$	27.79	83.66
B-Net	B-Net [20]	140.45	560.70
	B-Net^C	135.13	535.05
	DecomposeMe$_6$	56.19	271.52
	DecomposeMe$_5$	51.87	252.50
	DecomposeMe$_6^C$	63.89	289.20
	DecomposeMe$_5^C$	47.02	226.53
	DecomposeMe$_8^{C-256}$	38.54	130.13

(b)

Fig. 7. Time Analysis. a) Time as a function of L, see Eq. (2), and the size of the kernels, d. Time is measured as the total of a forward-backward pass of a convolutional layer for the baselines and two 1D convolutions and a rectifier linear unit for decomposed layers. Baselines correspond to convolutional layers using the same number of input-output 2D kernels. Feature map sizes vary corresponding to typical sizes in AlexNetOWT and B-Net. Times are obtained using a batch size of 32. b) Time benchmarks (in seconds) for the baselines and different instances of our proposed method. Timings are obtained using batches of 8 RGB images of size 224 × 224. Both timings are obtained on a Tesla K20m GPU.

ImageNet (Figure 5a), compact versions provide competitive results compared to the (full) baseline. Compact B-net models on ImageNet provide slightly lower performance than their equivalent full models as shon in Figure 5b. Nevertheless, the drop of performance is negligible. For these compact models we also observe in Figure 5c that the gap between training and validation accuracy is negative during most of the training process and, therefore, suggests that these models are too small for this particular dataset. The behavior of decomposed versions of the B-net structure on Places2 is different as shown in Figure 6b and Figure 6d. As summarized in Table 1b, all models provide similar performance on this dataset. These results suggest that 2D filters are, in fact, sub-optimal layers that need additional fully connected layers to improve performance. Compared to the baselines, compact models lead to an even more significant reduction in the total number of parameters, see Table 1. From these results, we can conclude that using 1D convolution layers not only reduces the number of operations and parameters, but also provides competitive (or better) performance compared to state-of-the-art methods.

The significant reduction in the number of parameters and memory footprint has not only benefits at test time. During training, these compact models make a better use of resources available. For instance, it is possible to increase the batch size to improve the estimation of gradients and, therefore, leverage larger amounts of data. The bottom line of Table 1b shows one additional instance of our method with larger number of decompositions trained with a batch size of 256, referred to as DecomposeMe$_8^{C-256}$. Please, note that this was not feasible using the baselines. As we can see, the number of parameters of this model is significantly lower than the baseline (e.g., 92% reduction) and, more importantly, there is a significant improvement in accuracy and computational complexity as we will see in Section 4.2.

4.2 Complexity analysis

Figure 7a shows the empirical computational costs of 2D convolutional layers (baselines) and decomposed layers for different representative layers. The plot represents the total time required in a forward-backward pass as a function of L. For the baseline, we report the time required solely for the convolution while for decomposed layers we report the combination of 1D convolution, non-linear layer, and 1D convolution. As we can see, the first layer does not produce any benefits time-wise. However, the significant reduction in time occurs for subsequent layers especially for using kernel sizes larger than 3×3. As shown, a more substantial reduction is achieved when L is similar to the number of input filters.

Empirical costs for baselines and instances of decompositions used in our experiments are summarized in Figure 7b. As expected, we can observe that the amount of time spent during fully connected layers is not meaningful compared to the time required by convolutional layers (see the comparison between AlexNetOWTBn and AlexNetOWTBn^C). Besides, substantial savings occur for instances of B-Net models where pairs of layers are decomposed and, therefore, maintaining the number of layers.

A fair comparison with existing low-rank approximation methods [2,4] is difficult as they require a fully pre-trained network to initialize their methods and, they need a fine-tuning process to prevent significant drops in performance. Contrary to them, our method is trained directly from data using a standard initialization. Compared to [2], for ImageNet considering AlexNetOWTBn as a similar network architecture (four convolutional layers and three fully connected layers), we obtain an increment in the top-1 performance of 5.87% with a $5.4x$ reduction in the number of weights. Our result is significantly better than the $2.5x$ reduction in the number of weights with an increment in error (top-5) of 0.02% reported in [2]. Best results reported in [4] are a speedup of $2.5x$ with no loss in accuracy and a $4.5x$ with a drop of 1% in classification accuracy on ICDAR2003. In our case, our best result is on Places2 where we achieve a $3.5x$ speedup in forward time with a $12.5x$ reduction in the number of parameters and an increment in top-1 classification accuracy of 5.7%.

4.3 Stereo Matching

The purpose of this experiment is to further demonstrate the applicability of our method when converting existing complex architectures. Accomplishing this, we address the problem of computing the disparity for each pixel in an image given a stereo pair of images. In particular, we use the recent method proposed in [21] where Zbontar *et al.* propose a ConvNet that matches patches in a stereo pair. The architecture consists of two feature extraction models, one per image and whose output serves as input for learning the matching network [21]. The entire process is learned in an end-to-end fashion and provides state-of-the-art results on KITTI2012 [38].

In this experiment, we focus on converting the feature extraction models to decomposed ones. These modules use four consecutive convolutional layers with kernels of size 3×3 with rectified linear units following each layer. Demonstrating the versatility of our architecture we test two different decompositions as outlined in Figure 2c. Firstly, we pair every two convolutional layers and transform them into a decomposed

Table 2. KITTI2012. a) Results for two instances of our method compared to the baseline. Al models, including the baseline, are retrained from scratch. Time is the forward-backward time over a batch of 8 images measured on a Tesla K20m. b) The leading submission on the KITTI 2012 leader-board as of 1st November 2015.

	Error	#FP[a]	#Lay	$\max d$	Runtime	Feat. Time
Baseline (retrained) [21]	2.60	339K	4	3	64.5s	776.9s
DecomposeMe[a]	2.66	48K	4	5	63.1s	312.9s
DecomposeMe[b]	2.72	32K	2	9	63.5s	281.9s

(a)

Method	ON[b]	OA	AN	AA
MC-CNN-acrt [21]	2.43	3.63	0.7 px	0.9 px
DecomposeMe[b]	2.48	3.69	0.8 px	0.9 px
MC-CNN	2.61	3.84	0.8 px	1.0 px

(b)

[a] FP stands for the number of parameters in convolutional layers. #Lay refers to the number of (1D or 2D) convolution layers. $\max d$ is the largest kernel size in the network.

[b] ON (Out-Noc) and OA (Out-All) stand for percentage of erroneous pixels in non-occluded areas and in total respectively. AN (Avg-Noc) and AA (Avg-All) stand for average disparity / end-point error in non-occluded areas and in total respectively.

one. Secondly, we consider a unique decomposition that compacts the four layers into a decomposed one using larger kernels of size 9×9. Therefore, both decompositions leverage the same neighborhood of size 9×9 in the input feature map which is the equivalent to four consecutive convolutions of 3×3 in the original model. Table 2 summarizes the results for these models. We show the numbers after retraining the original network (and, therefore, all randomization is equivalent). Table 2 includes the run time for the complete process including the matching network as well as the time required to extract features which is the focus of this experiment. As we can see, our approach significantly reduces the time required to extract features from each image. Nevertheless, this has almost no impact in the overall time which is consistent with the original paper [21], as the feature part is not responsible for the majority of the computational cost. More importantly, our proposed method achieves almost the same performance with a significant reduction in the number of parameters.

Finally, Table 2b summarizes bench-marking results on KITTI dataset [38]. Our proposal provides similar results compared to the original network using only 24.3% of the parameters in the feature layers. These are relevant results since our proposed method, without a custom design, can reach similar performance compared to a deep model that was carefully engineered. More importantly, this is achieved using only a fraction of the number of parameters.

5 Conclusions

In this paper we proposed DecomposeMe. A novel and efficient convolutional neural network architecture based on 1D convolutions. Experiments on large-scale image classification show that our approach improves the classification accuracy while significantly reducing the number of parameters and computational cost. For instance, on Places2 and compared to the VGG-B model, our architecture obtains a relative improvement in top-1 classification accuracy of 7.7% using 92% fewer parameters than VGG-B and with a speed up factor in forward-time of $3.5x$. Additional experiments on stereo matching also demonstrate the general applicability of the proposed architecture.

Acknowledgment The authors thank John Taylor for helpful discussions and continuous support through using the CSIRO high-performance computing facilities. The authors also thank NVIDIA for generous hardware donations.

References

1. P. Sermanet, D. Eigen, X. Zhang, M. Mathieu, R. Fergus, and Y. LeCun, "Overfeat: Integrated recognition, localization and detection using convolutional networks," in *International Conference on Learning Representations (ICLR2014)*. CBLS, April 2014.
2. E. L. Denton, W. Zaremba, J. Bruna, Y. LeCun, and R. Fergus, "Exploiting linear structure within convolutional networks for efficient evaluation," in *NIPS*, 2014, pp. 1269–1277.
3. A. Sironi, B. Tekin, R. Rigamonti, V. Lepetit, and P. Fua, "Learning separable filters," *PAMI*, vol. 37, no. 1, pp. 94 – 106, 2015.
4. M. Jaderberg, A. Vedaldi, and A. Zisserman, "Speeding up convolutional neural networks with low rank expansions," in *British Machine Vision Conference*, 2014.
5. G. F. Montufar, R. Pascanu, K. Cho, and Y. Bengio, "On the number of linear regions of deep neural networks," in *Advances in Neural Information Processing Systems 27*, Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, and K. Weinberger, Eds., 2014, pp. 2924–2932.
6. A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems*, 2012.
7. K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *CoRR*, vol. abs/1409.1556, 2014.
8. V. O. Hinton, G. E. and J. Dean, "Distilling the knowledge in a neural network." in *arXiv*, 2014.
9. M. Denil, B. Shakibi, L. Dinh, M. Ranzato, and N. de Freitas, "Predicting parameters in deep learning," *CoRR*, vol. abs/1306.0543, 2013.
10. M. Jaderberg, A. Vedaldi, and A. Zisserman, "Speeding up convolutional neural networks with low rank expansions," vol. abs/1405.3866, 2014.
11. B. Liu, M. Wang, H. Foroosh, M. Tappen, and M. Penksy, "Sparse convolutional neural networks," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2015.
12. R. Rigamonti, A. Sironi, V. Lepetit, and P. Fua, "Learning separable filters," in *Conference on Computer Vision and Pattern Recognition*, 2013.
13. E. L. Denton, W. Zaremba, J. Bruna, Y. LeCun, and R. Fergus, "Exploiting linear structure within convolutional networks for efficient evaluation," in *NIPS*, 2014, pp. 1269–1277.
14. A. Krizhevsky, "One weird trick for parallelizing convolutional neural networks," *CoRR*, vol. abs/1404.5997, 2014. [Online]. Available: http://arxiv.org/abs/1404.5997
15. Y. LeCun, "Lenet-5, convolutional neural networks," 2015. [Online]. Available: http://yann.lecun.com/exdb/lenet/
16. J. Snoek, H. Larochelle, and R. P. Adams, "Practical bayesian optimization of machine learning algorithms," in *Advances in Neural Information Processing Systems*, 2012.
17. Y. LeCun and C. Cortes, "MNIST handwritten digit database," 2010.
18. A. Krizhevsky and G. Hinton, "Learning multiple layers of features from tiny images," *Technical Report*, 2009.
19. O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. S. Bernstein, A. C. Berg, and F.-F. Li, "Imagenet large scale visual recognition challenge." *CoRR*, vol. abs/1409.0575, 2014.
20. B. Zhou, A. Khosla, A. Lapedriza, A. Torralba, and A. Oliva, "Places2: A large-scale database for scene understanding," 2015.
21. J. Zbontar and Y. LeCun, "Stereo matching by training a convolutional neural network to compare image patches," *CoRR*, vol. abs/1510.05970, 2015.
22. A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? the kitti vision benchmark suite," in *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012.

23. R. Collobert, K. Kavukcuoglu, and C. Farabet, “Torch7: A matlab-like environment for machine learning,” in *BigLearn, NIPS Workshop*, 2011.
24. GitHub. (2015) soumith/imagenet-multigpu.torch. [Online]. Available: https://github.com/soumith/imagenet-multiGPU.torch
25. M. Lin, Q. Chen, and S. Yan, “Network in network,” *CoRR*, vol. abs/1312.4400, 2013.
26. J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and F.-F. Li, “Imagenet: A large-scale hierarchical image database.” in *CVPR*, 2009, pp. 248–255.
27. S. Ioffe and C. Szegedy, “Batch normalization: Accelerating deep network training by reducing internal covariate shift,” *CoRR*, 2015.
28. X. Glorot and Y. Bengio, “Understanding the difficulty of training deep feedforward neural networks,” in *JMLR W&CP: Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics (AISTATS 2010)*, vol. 9, May 2010, pp. 249–256.
29. S. R. Kaiming He, Xiangyu Zhang and J. Sun, “Delving deep into rectifiers: Surpassing human-level performance on imagenet classification.” in *ICCV*, 2015.
30. A. Vedaldi and K. Lenc, “Matconvnet - convolutional neural networks for MATLAB,” *CoRR*, vol. abs/1412.4564, 2014.
31. C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, “Googlelenet: Going deeper with convolutions,” in *Computer Vision and Pattern Recognition*, vol. 2, June 2015.
32. K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” *CoRR*, vol. abs/1512.03385, 2015. [Online]. Available: http://arxiv.org/abs/1512.03385
33. in *ICLR workshops*.
34. G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, “Improving neural networks by preventing co-adaptation of feature detectors,” *CoRR*, vol. abs/1207.0580, 2012. [Online]. Available: http://arxiv.org/abs/1207.0580
35. R. G. L. Z. Michael Cogswell, Faruk Ahmed and D. Batra, “Reducing overfitting in deep networks by decorrelating representations.” in *ICLR*, 2016.
36. E. Denton, W. Zaremba, J. Bruna, Y. LeCun, and R. Fergus, “Exploiting linear structure within convolutional networks for efficient evaluation,” in *NIPS*, 2014.
37. A. Krizhevsky and G. Hinton, “Learning multiple layers of features from tiny images,” 2009.
38. A. Geiger, P. Lenz, C. Stiller, and R. Urtasun, “Vision meets robotics: The kitti dataset,” *International Journal of Robotics Research (IJRR)*, 2013.