Reborn Filters: Pruning Convolutional Neural Networks with Limited Data

Yehui Tang,[1*] **Shan You,**[2,6*] **Chang Xu,**[3*] **Jin Han,**[1] **Chen Qian,**[2] **Boxin Shi,**[4,5*] **Chao Xu,**[1] **Changshui Zhang**[6]
[1]Key Lab of Machine Perception (MOE), CMIC, School of EECS, Peking University, China
[2]SenseTime Research, [3]School of Computer Science, Faculty of Engineering, The University of Sydney, Australia
[4]National Engineering Laboratory for Video Technology, Peking University [5]Peng Cheng Laboratory
[6]Department of Automation, Tsinghua University
{yhtang, hanjin1619, shiboxin}@pku.edu.cn, {youshan, qianchen}@sensetime.com,
c.xu@sydney.edu.au, xuchao@cis.pku.edu.cn, zcs@mail.tsinghua.edu.cn

Abstract

Channel pruning is effective in compressing the pretrained CNNs for their deployment on low-end edge devices. Most existing methods independently prune some of the original channels and need the complete original dataset to fix the performance drop after pruning. However, due to commercial protection or data privacy, users may only have access to a tiny portion of training examples, which could be insufficient for the performance recovery. In this paper, for pruning with limited data, we propose to use all original filters to directly develop new compact filters, named reborn filters, so that all useful structure priors in the original filters can be well preserved into the pruned networks, alleviating the performance drop accordingly. During training, reborn filters can be easily implemented via 1×1 convolutional layers and then be fused in the inference stage for acceleration. Based on reborn filters, the proposed channel pruning algorithm shows its effectiveness and superiority on extensive experiments.

Introduction

There are always strict limits on memory and inference speed of deep convolutional neural networks (CNNs) deployed on various low-end edge devices (*e.g.*, mobile phones, tablets and wearable gadgets) (Cheng et al. 2017; Wang, Li, and Smola 2019). However, popular CNN models such as VGGNet (Simonyan and Zisserman 2014) and ResNet (He et al. 2016) have a huge number of model parameters and fail to meet the demands (Kim, Park, and Kwak 2018; Louizos, Ullrich, and Welling 2017; Wang et al. 2016; You et al. 2017b).

Model compression serves as a remedy by investigating and further eliminating the redundancy in the networks (Alvarez and Salzmann 2017; You et al. 2017a). For example, quantization (Courbariaux, Bengio, and David 2015; Rastegari et al. 2016; Li et al. 2017) attempts to reduce the redundancy on numerical precision, and sparse connection (Han, Mao, and Dally 2015; Wen et al. 2016; Alvarez and Salzmann 2016) or tensor factorization (Lebedev et al. 2014; Novikov et al. 2015) seek to make the parameters to be sparse or low-rank. However, most of them (*e.g.*, (Han, Mao, and Dally 2015)) may need special hardwares for real acceleration. In contrast, prevalent channel-pruning methods tend to identify and eliminate redundant channels, which lead to a significant decrease of the size of feature maps and CNN filters. Most importantly, network structure of the pruned network will not be changed, so it is friendly for the off-the-shelf deep learning frameworks and complementary to other compression methods.

In channel pruning, one of the mainstream practices is to prune the channels (*i.e.* removing the corresponding filters) according to the layer-wise reconstruction error (He, Zhang, and Sun 2017; Luo, Wu, and Lin 2017). Most methods just evaluate the importance of filters channel by channel, and remove them for good. Nevertheless, this brings in the elimination of redundant information as well as the risk of discarding some useful individual information, which usually induces large reconstruction error and the common *catastrophic drop* phenomenon on the network performance after the pruning. Therefore, most methods need to fine-tune the pruned network over sufficient data (ideally the complete original dataset) to recover the performance. It has been empirically found that even the training-from-scratch method (Liu et al. 2018) is also capable of recovering the performance, when all training data sample are used.

However, in many real-world applications, users may not have the access to sufficient or complete data used for training the original networks. This scenario is rather practical when it comes to some considerations such as commercial interests or user's experience. For example,

- Powerful models trained on fairly large datasets are released while the dataset is not available due to commercial profits. (Mahajan et al. 2018) trained CNNs on 940 million of Instagram images and released pre-trained models[1], whose accuracy on image recognition task was much higher than those trained on public datasets. However, only some example images can be found, and the training dataset will not be released as mentioned in the paper.
- For the model compression service provided by the cloud, asking consumers to upload the whole huge training

*Corresponding authors

[1]https://github.com/facebookresearch/WSL-Images

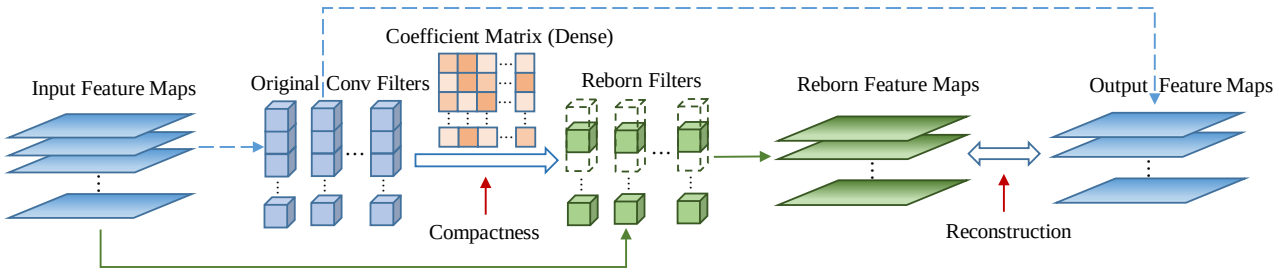


Figure 1: Pruning pipeline with reborn filters. Dashed blue line for original networks and solid green line for pruned networks. Reborn filters are developed by refining all the original convolutional filters in a layer, and making their input channels sparse (dashed green cubes) will induce pruning the input feature maps.

dataset (*e.g.* ImageNet with over 120GB file size) with limited transmission speed (*e.g.* 20M/s) is unpractical, which takes a long time and does harm user's experience. Instead, the cloud only needs the trained models (only several MB parameters, *e.g.* 45 MB for Resnet-18) and very few sample data to complete the compression task.

With these limited data, the effect of both fine-tuning and training from scratch will be restricted, and they can not recover the missed information after pruning especially for a large pruning ratio. Then the ultimate performance of the compressed networks is mainly up to the extent of catastrophic drop before fine-tuning.

In this paper, we reduce the information loss of original networks via refining all the original filters during pruning to achieve the goal of channel pruning using limited data and directly alleviate the performance drop. Instead of exploring the individual redundancy for each channel, we examine it from a group (layer) perspective, and all filters in a layer are used to develop new compact filters, named *reborn filters* as Figure 1 shows. In this way, the information of the original filters, which guarantees the performance, are well preserved in the reborn filters, thus the performance drop will be minimized accordingly. These structure priors from original filters help the limited data to learn compact reborn filters for pruning. Concretely, we assume at a certain layer, each input channel of a reborn filter is constructed as a linear representation across all input channels of an original filter to inherit the discovered patterns or structures. In a layer-wise manner, reborn filters are used to reconstruct the feature maps of examples under a compactness constraint, *i.e.*, with sparse input channels. Then we implement pruning to the zero channels of the learned reborn filters.

Moreover, reborn filters are easy to implement by 1×1 convolutional layers with linear activations, which are imposed right before the traditional CNN layers as Figure 2 shows later. During the test, 1×1 convolutional layers can be fused with traditional CNN layers for acceleration. We have conducted extensive experiments on the benchmark CIFAR-10 dataset and the large-scale ImageNet dataset. The results demonstrate the effectiveness and superiority of our proposed method.

Formulation of Reborn Filters

In this section, we first introduce definition of the proposed reborn filters and elaborate how they are constructed by the original filters. Then we discuss how compact reborn filters relate to the channel pruning. Without loss of generality, we focus on the CNN filters at a certain layer, and we aim to prune the channels of its input feature maps as (He, Zhang, and Sun 2017; Luo, Wu, and Lin 2017; Zhuang et al. 2018).

Reborn filters

For a pre-trained CNN model (*e.g.*, VGGNet and ResNet), its convolutional filters indicate our understanding of intrinsic patterns underlying original training data, and are the key components for its powerful ability. At a certain layer [2], the convolutional filters of the pre-trained model can be denoted as a 4-dimensional tensor $\mathcal{W} \in \mathbb{R}^{n \times m \times d_h \times d_w}$, where n and m are the number of output channels and input channels, respectively; d_h and d_w are the window size of the convolution, *e.g.*, 3×3. Then $\mathcal{W}_{i,j,:,:}$ represents the 2-dimensional convolutional filter for the i-th output channel and j-th input channel.

The various $\mathcal{W}_{i,j,:,:}$'s in pre-trained model extract informative patterns and structures, such as some low-level edges and semantic-related textures. These learned filters are supposed to be highly relevant with the task and contribute to the performance. Since in model compression, the pruned network has a lower network capacity than that of the original network. To preserve the original recognition performance, a natural idea is to inherit these well trained filters. For each output channel i, we formulate a reborn filter as a linear combination of its corresponding input channels, *i.e.*, for reborn filter $\tilde{\mathcal{W}}_{i,j,:,:}$,

$$\tilde{\mathcal{W}}_{i,j,:,:} = \sum_{k=1}^{m} a_{kj} \mathcal{W}_{i,k,:,:}, \quad \forall i \in [1:n], j \in [1:m], \quad (1)$$

where a_{kj} is the element of matrix $A \in \mathbb{R}^{m \times m}$ in the k-th row and the j-th column. In the sequel, we use the mark tilde $\tilde{}$ over a symbol to indicate the feature maps or convolutional filters related to the reborn filters. In this way, reborn filters can inherit the knowledge embedded in original convolutional filters that have been trained for days or weeks. This practice coheres with the idea of multiple kernel learning (MKL) (Gönen and Alpaydın 2011; Bucak, Jin, and Jain 2014), which seeks to learn an optimal combination of pre-defined kernels (aka the original filters),

[2] For simplicity of illustration, we omit the layer index since our reborn filters fit for all layers.

so that we can adequately exploit the knowledge within the already-learned filters.

Compactness of the reborn filters

In channel pruning, most methods follow a selecting strategy, *i.e.*, directly selecting some original filters via weighting (He, Zhang, and Sun 2017), scaling factors in batch normalization layers (Liu et al. 2017) or greedy selection (Luo, Wu, and Lin 2017; Zhuang et al. 2018). In contrast, we propose to directly develop new and compact reborn filters using all the original filters, so that the useful information can be actually maintained. And those redundant ones can be pruned naturally, making both the model size and calculation complexity decreased.

Concretely, we denote the vector-form of original filter $\mathcal{W}_{i,k,:,:}$ as $\boldsymbol{f}_{ik} = \text{vec}(\mathcal{W}_{i,k,:,:}) \in \mathbb{R}^{d_h d_w}$, where $\text{vec}(\cdot)$ is the vectorization of matrixes according to columns. Then all m filters w.r.t., the i-th output channel is $F_i = [\boldsymbol{f}_{i1}, \boldsymbol{f}_{i2}, ..., \boldsymbol{f}_{im}] \in \mathbb{R}^{d_h * d_w \times m}$, and Eq. (1) can be equivalently written into

$$\tilde{\boldsymbol{f}}_{ij} = \text{vec}(\tilde{\mathcal{W}}_{i,j,:,:}) = \sum_{k=1}^{m} a_{kj} \boldsymbol{f}_{ik} = F_i \boldsymbol{a}_j. \tag{2}$$

Letting $\tilde{F}_i = [\tilde{\boldsymbol{f}}_{i1}, \tilde{\boldsymbol{f}}_{i2}, ..., \tilde{\boldsymbol{f}}_{im}]$, we also have $\tilde{F}_i = F_i A$. By denoting a filter matrix F as

$$F = [F_1^T, F_2^T, ..., F_n^T]^T = [\boldsymbol{f}_1, ..., \boldsymbol{f}_m] \in \mathbb{R}^{n * d_h * d_w \times m},$$

we have all reborn filters $\tilde{F}$ represented by the original filters F and the coefficient matrix A, *i.e.*, $\tilde{F} = FA$, each column of which corresponds to the filters for each input channel. Then for compact reborn filters w.r.t. input channels, we encourage the reborn filter matrix $\tilde{F}$ to have sparse columns, which is reflected by its $\ell_{2,0}$-norm, *i.e.*,

$$\left\| \tilde{F} \right\|_{2,0} = \|FA\|_{2,0} = \sum_{i=1}^{m} \mathcal{B}(\|F\boldsymbol{a}_i\|_2 > 0), \tag{3}$$

where $\mathcal{B}(\cdot)$ is the Boolean indicator function.

Remark. Note that current works (Luo, Wu, and Lin 2017; Zhuang et al. 2018; He, Zhang, and Sun 2017) exactly remove some columns of F for good, which might cause information loss and this loss can not be compensated by the limited data. In contrast, we construct compact $\tilde{F} = FA$ using F, so that the complete information about F can be well preserved in $\tilde{F}$. And the redundant filters can be automatically obtained by mutual determination of all channels (columns), instead of being identified harshly via selection. As a result, the performance drop can be minimized, which benefits the limited data case. [3]

Channel Pruning with Reborn Filters

Reborn filters inherit the information of all original filters. Then feature maps are expected to be reconstructed with smaller errors, which is the key to ensure the performance of the pruned network with limited data. Now we formally present our channel pruning approach based on reborn filters. Our solution also consists of three procedures: first, reborn filters are used to reconstruct its output feature maps with sparse input channels, then redundant input channels of reborn filters are pruned as well as the corresponding input feature maps. Finally the labels are used to fine-tune the pruned network slightly, but we suggest that it will not be necessary if the data are quite limited since the improvement will be fairly small.

Layer-wise reconstruction

At a certain layer, given N examples $\{\boldsymbol{x}_i\}_{i=1}^N$ and the original filter $\mathcal{W} \in \mathbb{R}^{n \times m \times d_h \times d_w}$, the corresponding input feature maps and output feature maps are denoted as two 4-dimensional tensors $\boldsymbol{X} \in \mathbb{R}^{N \times m \times h \times w}$ and $\boldsymbol{Z} \in \mathbb{R}^{N \times n \times h' \times w'}$, respectively, where $h \times w$ and $h' \times w'$ are corresponding feature map size. Then for the t-th example $\boldsymbol{x}_t$, the i-th channel of the output tensor $\boldsymbol{Z}$, *i.e.*, $\boldsymbol{Z}_{t,i,:,:} \in \mathbb{R}^{h' \times w'}$, is calculated via the convolution of the input tensor $\boldsymbol{X}_{t,:,:,:} \in \mathbb{R}^{m \times h \times w}$ and the corresponding filters $\mathcal{W}_{i,:,:,:} \in \mathbb{R}^{m \times d_h \times d_w}$, *i.e.*,

$$\boldsymbol{Z}_{t,i,:,:} = \sum_{j=1}^{m} \mathcal{W}_{i,j,:,:} * \boldsymbol{X}_{t,j,:,:}, \tag{4}$$

where $*$ is the 2D convolutional operator. In this way, the output tensor with reborn convolutional filters is thus

$$\tilde{\boldsymbol{Z}}_{t,i,:,:} = \sum_{j=1}^{m} \tilde{\mathcal{W}}_{i,j,:,:} * \boldsymbol{X}_{t,j,:,:} \tag{5}$$

$$= \sum_{j=1}^{m} \left(\sum_{k=1}^{m} a_{kj} \mathcal{W}_{i,k,:,:} \right) * \boldsymbol{X}_{t,j,:,:} \tag{6}$$

$$= \sum_{j=1}^{m} \sum_{k=1}^{m} a_{kj} (\mathcal{W}_{i,k,:,:} * \boldsymbol{X}_{t,j,:,:}). \tag{7}$$

Then to maintain the network performance of the original network, reborn filters are supposed to reconstruct the original output feature maps by minimizing the reconstruction error measured by the mean squared error (MSE) between two output feature maps, *i.e.*,

$$\ell_{re}(A; \boldsymbol{x}_t) = \frac{1}{2} \sum_{i=1}^{n} \left\| \tilde{\boldsymbol{Z}}_{t,i,:,:} - \boldsymbol{Z}_{t,i,:,:} \right\|_F^2, \tag{8}$$

where $\|\cdot\|_F$ is the Frobenius norm of a matrix or a vector. And the reconstruction error over all examples is

$$\mathcal{L}_{re}(A; \mathcal{D}) = \frac{1}{N} \sum_{t=1}^{N} \ell_{re}(A; \boldsymbol{x}_t)$$

$$= \frac{1}{2N} \sum_{t=1}^{N} \sum_{i=1}^{n} \left\| \boldsymbol{Z}_{t,i,:,:} - \sum_{k,j=1}^{m} a_{kj} \boldsymbol{O}_{t,i,k,j} \right\|_F^2, \tag{9}$$

where $\boldsymbol{O}_{t,i,k,j} = \mathcal{W}_{i,k,:,:} * \boldsymbol{X}_{t,j,:,:} \in \mathbb{R}^{h' \times w'}$ is a fixed data matrix. Following Eq. (3), in channel pruning besides the reconstruction ability, we aim to encourage the reborn filters

[3] In the following, we seamlessly use the filter notations $\mathcal{W}$ and its vector form F. The choice depends on the simplicity of mathematical expressions.

to be sparse over the input channels. Thus for a predefined pruning rate $r \in (0, 1)$ (Alvarez and Salzmann 2016; Liu et al. 2017), we cast the problem as follows,

$$\min_{A} \mathcal{L}_{re}(A; \mathcal{D}) \quad \text{s.t. } \|FA\|_{2,0} \leq 1 - \lceil rm \rceil, \tag{10}$$

where $\lceil rm \rceil$ is the largest integer no larger than rm. However, solve this ℓ_{20} minimization problem Eq. (10) is NP-hard. Hence, we adopt a sparsity-induced surrogate ℓ_{21} norm, *i.e.*, $\|FA\|_{2,1} = \sum_{i=1}^{m} \|F\boldsymbol{a}_i\|_2$, and relax problem Eq. (10) into the minimizing the objective

$$\mathcal{L}_{tr}(A; \mathcal{D}) = \frac{1}{N} \sum_{t=1}^{N} \ell_{re}(A; \boldsymbol{x}_t) + \lambda \|FA\|_{2,1}, \tag{11}$$

where $\lambda > 0$ is a constant to control the sparsity. From the optimization perspective, Eq. (11) is a group fused lasso problem (Simon et al. 2013), which can be solved by many off-the-shelf convex toolboxes, such as CVX (Grant and Boyd 2014) and SPAMS (Mairal et al. 2010). In our paper, we adopt the widely-used alternating direction method of multipliers (ADMM) algorithm (Boyd et al. 2011) since it enjoys fast convergence rate $\mathcal{O}(1/T)$ with the number of iterations T. In particular, to save the computation cost, we choose a fast stochastic version (Zhong and Kwok 2014) of ADMM while having the same convergence rate as the traditional batch ADMM. Details refer to the supplementary materials.

Pruning and fine-tuning

After solving Eq. (11), the matrix FA will have sparse columns. Denote the index set of its non-zero columns as $\Omega \subset [1 : m]$ and its complementary set as $\bar{\Omega} = [1 : m] - \Omega$. Then we just prune the input channels of reborn filters according to the index set $\bar{\Omega}$. After obtaining the compact reborn filters, we can fine-tune the remaining reborn filters slightly to further reduce the reconstruction error as well as accommodate the new layer structure as (He, Zhang, and Sun 2017; Zhuang et al. 2018), and then move on to the next layer until the final softmax layer is reached. The obtained network is denoted as $\tilde{\mathcal{N}}_{\Omega}$. Finally we can use the labels to fine-tune the whole pruned network $\tilde{\mathcal{N}}_{\Omega}$ via a regular cross-entropy loss for further boosting the classification performance. Nevertheless, we empirically find that the final fine-tuning has little effect on the performance improvement when the training data are quite limited. By controlling the reconstruction error layer-wisely, the original network provides additional strict supervision signals for the pruned network, enabling the classification performance to be inherited as much as possible. In this sense, our approach can also handle the case when ground-truth labels of the limited training data are unknown.

Reborn filters and 1×1 convolution

As illustrated before, reborn filters are developed by the linear combination of the convolutional filters from the original network. Now we proceed from a different perspective, and suggest that reborn filters can be easily realized by the widely-used 1×1 convolution, which benefits the implementation of reborn filters in various real-world applications.

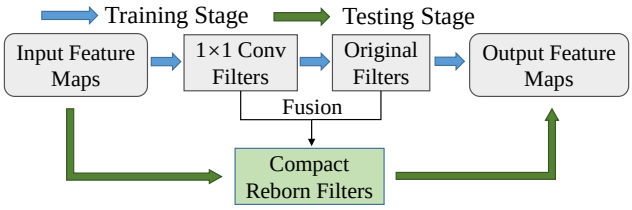


Figure 2: Using 1×1 convolution to implement the reborn filters.

By rewriting Eq. (5), we have

$$\tilde{\boldsymbol{Z}}_{t,i,:,:} = \sum_{j=1}^{m} \sum_{k=1}^{m} a_{kj} (\mathcal{W}_{i,k,:,:} * \boldsymbol{X}_{t,j,:,:}) \tag{12}$$

$$= \sum_{k=1}^{m} \mathcal{W}_{i,k,:,:} * (\sum_{j=1}^{m} a_{kj} \boldsymbol{X}_{t,j,:,:}) \tag{13}$$

$$= \sum_{k=1}^{m} \mathcal{W}_{i,k,:,:} * (A_{k,:} \circledast \boldsymbol{X}_{t,:,:,:}), \tag{14}$$

where $A_{k,:}$ indicates the k-th row of matrix A, and $\circledast$ stands for the 1×1 convolution. As a result, the normal convolution using reborn filters is decomposed into two procedures: first, we implement 1×1 convolution on the input feature maps using each of A's row as the weights, with m new input channels obtained; second, we implement the normal convolution on the new input feature maps using the original filters.

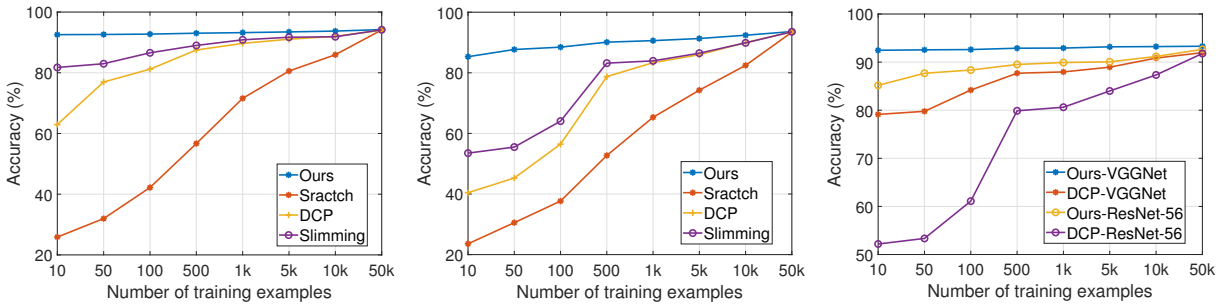
The diagram of using 1×1 convolution to implement reborn filters is shown in Figure 2. As we can see, the reborn filters are equivalently m 1×1 convolutional layers imposed before the input feature maps with linear activations. During training, the layer-wise reconstruction can be viewed as a fairly *shallow* network with the original filters fixed. This perspective enables us to fastly calculate the gradients over A in optimizing ADMM (see line 6 of Algorithm 1 in supplementary materials) by dint with the BP algorithm. Furthermore, when pruning the network, we can simply let those 1×1 convolutional weights corresponding to the pruned channels be zero and fixed. Note that for reborn filters, we do not necessarily need the 1×1 convolutional layer to be sparse; actually, dense 1×1 filters and dense original filters can also develop compact reborn filters, which is determined automatically by exploring the group redundancy of all original filters. As for the test stage, to obtain a real pruned network, we need to eliminate those 1×1 convolutional layers by fusing them into the ordinary convolutional layers as reborn filters Eq. (1). The use of 1×1 convolution can also refer to (Li et al. 2018); however, the difference lies in that (Li et al. 2018) adopts 1×1 convolution as a manner for knowledge distillation in a teacher-student paradigm while we use 1×1 convolution to develop new compact filters from original filters for channel pruning.

Experimental Results

In this section, we empirically investigate the performance of our introduced reborn filters and proposed algorithm. We

Table 1: Top-1 accuracy and compression & acceleration rates of the pruned VGGNet and ResNet-56 with $N = 500$ (1%) training examples on CIFAR-10 dataset. The testing accuracies of pre-trained VGGNet and ResNet-56 are 93.99% and 93.80% respectively.

Model	VGGNet				ResNet-56			
Method	Slimming	DCP	Scratch	Ours	Slimming	DCP	Scratch	Ours
# Training data	500	500	500	500	500	500	500	500
#Param ↓	$\sim 6\times$	$\sim 2\times$	$\sim 6\times$	$\sim 6\times$	$\sim 2\times$	$\sim 2\times$	$\sim 2\times$	$\sim 2\times$
#FLOPs ↓	$\sim 2\times$	$\sim 2\times$	$\sim 2\times$	$\sim 2\times$	$\sim 2\times$	$\sim 2\times$	$\sim 2\times$	$\sim 2\times$
Accuracy (%)	87.51±0.54	89.94±0.19	56.69±1.31	**93.03±0.19**	78.74±0.86	83.19±0.47	52.74±1.15	**90.11±0.45**



(a) VGGNet (b) ResNet-56 (c) Without final fine-tuning

Figure 3: Accuracy drop of the pruned VGGNet and ResNet-56 on CIFAR-10 dataset w.r.t. different number N of training examples, with about $2\times$ #FLOPs ↓ for both VGGNet and ResNet-56.

select state-of-the-art end-to-end training-based Slimming method (Liu et al. 2017) and reconstruction-based DCP (Discrimination-aware Channel Pruning) method (Zhuang et al. 2018)) as our comparison methods. To show the baseline performance when the training data are limited, we also train the pruned network from scratch by randomly initializing the weights, which is denoted as "Scratch". For fairness of the comparison, we implement pruning the VGGNet and ResNet on the benchmark CIFAR-10 and ImageNet dataset since comparison methods (Liu et al. 2017) and (Zhuang et al. 2018) have been optimized to achieve state-of-the-art performance on them.

Implementation details. In our experiment, we randomly sample N images from the training set for pruning, and vary N to analyze how the number of training examples influences network compression. For CIFAR-10 dataset, we use the original testing set for testing while for ImageNet dataset, the original validation images are used. During training, the default value of batchsize is 128, and random cropping and mirror are used for augmentation. The sparsity weight λ in Eq.(11) is selected from $\{0.5, 0.4, 0.3, 0.2, 0.1, 0.05\}$ to obtain reborn filters with different sparsity, then filters with norm less than a threshold (set as 0.0005) are pruned, resulting in different acceleration rates. Details of the optimizers (ADMM and SGD) on two datasets refer to the supplementary materials. The proposed method is implemented with Pytorch (Paszke et al. 2017) on NVIDIA 1080 Ti GPUs.

Experiments on CIFAR-10

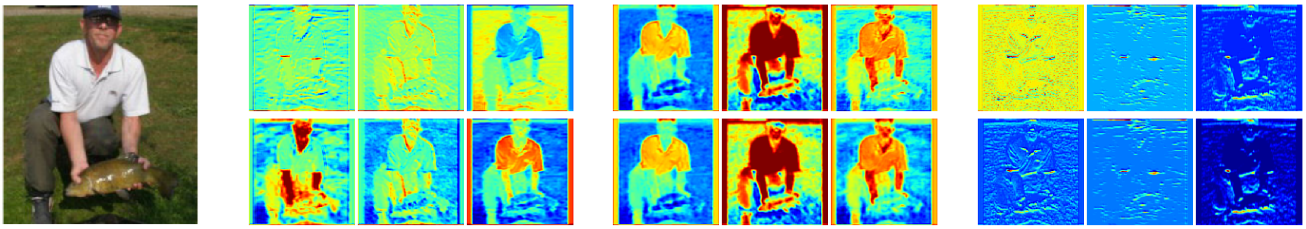
Dataset and pre-trained networks. The CIFAR-10 dataset consists of 60,000 RGB images from ten exclusive categories with size 32×32. Among these images, 50,000 of them are for training while the remaining 10,000 images are for testing.The testing accuracies of pre-trained VGGNet and ResNet-56 are 93.99% and 93.80% respectively. Besides, VGGNet has 20M parameters and 399M FLOPs while ResNet-56 has 0.86M parameters and 126M FLOPs.

Results. We sample only 1% images ($N = 500$) as the training set for pruning, and the obtained accuracies on testing dataset and compression statistics are shown in Table 1. All the methods are run five times with different random seeds. We can see with similar acceleration rate and even higher compression rate, our method significantly outperforms other comparison methods. For VGGNet, with high compression rate $6\times$ and acceleration rate $2\times$, our method can achieve accuracy 93.03% though the training data are quite limited. Note that the training-from-scratch method only achieves accuracy of 56.69% due to the limited data, which indicates the necessary to inherit the original filters. ResNet-56 is more challenging than VGGNet; nevertheless, with about $2\times$ compression(0.4M) and acceleration(63M), our pruned networks still have acceptable classification accuracy (90.11%), while those pruned by other methods all fail to meet the accuracy demands for real applications (less than 83.19%).

To comprehensively investigate the superiority of our method with respect to different number of training examples, we conduct pruning with increasing training examples N, and the accuracy is presented in Figure 3. From Figure 3 (a) and (b), we can see that our method enjoys huge superiority over other comparison methods when the training data are limited. We also report the accuracy without the final fine-tuning in Figure 3 (c). It can be seen that with quite limited data (*e.g.*, $<$ 5k), the effect of final fine-tuning is slight, and the performance before the final fine-tuning dominates the performance of the pruned networks. When the channels are pruned layer-wisely, our method can exactly ensure a nice accuracy by reducing the information loss and using the original structure priors. This also indicates in a way the ground-truth labels are not necessarily needed for

Table 2: Top-5 accuracy (%) and acceleration rates of the pruned VGGNet-16 and ResNet-50 with $N = 1k$ training examples ($< 1‰$) on ImageNet dataset.The top-5 accuracies of pre-trained VGGNet-16 and ResNet-50 are 91.50% and 92.87% respectively.

Model	Performance	Slimming	DCP	Scratch	Ours
VGGNet-16	# Training data	1000	1000	1000	1000
	#FLOPs ↓	$\sim 2\times$	$\sim 2\times$	$\sim 2\times$	$\sim 2\times$
	Accuracy(%)	42.60±0.12	85.27±0.03	6.67±0.85	**89.40±0.03**
ResNet-50	# Training data	-	1000	2.00 ×	1000
	#FLOPs ↓	-	$\sim 2\times$	$\sim 2\times$	$\sim 2\times$
	Accuracy(%)	-	82.85±0.04	6.51±0.96	**90.03±0.05**



(a) input image (b) "newly constructed" (c) "exactly inherited" (d) redundant

Figure 4: Visualization of feature maps w.r.t. learned reborn filters for the VGGNet-16 on ImageNet dataset. The top row indicates the original filters while the bottom row is for reborn filters.

Table 3: Top-5 accuracy (%) of the pruned VGGNet-16 on ImageNet dataset with $2\times$ #FLOPs ↓.The top-5 accuracy of pre-trained VGGNet-16 is 91.50%.

	N	Slimming	DCP	Scratch	Ours
VGGNet-16	100	25.08	78.74	2.58	**88.73**
	1k	42.6	85.27	6.67	**89.40**
	5k	58.02	87.38	12.34	**89.52**
	10k	68.18	87.76	20.05	**89.69**
	100k	82.77	88.3	74.1	**90.04**
	All	91.57	91.68	91.46	**91.75**

Table 4: Top-5 accuracy drop (%) of the pruned ResNet-18 on ImageNet dataset with $1.5\times$ #FLOPs ↓. The top-5 accuracy of pre-trained ResNet-18 is 89.08%.

	N	DCP	Scratch	Ours
ResNet-18	100	67.74	1.92	**84.11**
	1k	78.12	6.63	**84.84**
	5k	80.63	10.63	**85.14**
	10k	82.05	21.62	**85.32**
	100k	83.24	68.50	**85.67**
	All	88.73	88.65	**88.89**

our method when the training data are quite limited. With increasing training examples, the performance improves accordingly. Note that when the training data are sufficient (50k), the performance of pruned networks tend to be similar for all methods, which further validates that for a given "pruned" network, its performance is up to inheriting the original network when the training data are limited; otherwise, the performance is mainly dominated by the training data. Moreover, since reborn filters maintain the original filters, our method is more steady w.r.t. the number of training examples. Specially for VGGNet with only 10 examples, our method still can achieve accuracy 92.56%. In contrast, the accuracies of other comparison methods are up to 81.75%.

Experiments on ImageNet (ILSVRC2012)

Dataset and pre-trained networks. The ImageNet (ILSVRC2012) dataset (Russakovsky et al. 2015) is composed of 1.28 million training images and 50k validation images from 1000 categories. The widely-used VGGNet-16, ResNet-18 and ResNet-50 released by Pytorch [4] are adopted as pre-trained CNNs, whose single-view top-5 error are 91.50%, 89.08% and 92.87%, respectively. Besides, VGGNet-16, ResNet-18 and ResNet-50 respectively have 15.5B,1.8B and 4.1B FLOPs.

Results. For Table 2, with different random seeds, we randomly sample 1k ($< 1‰$) training images and run all the methods 3 times. Although the very low accuracy of training from scratch (less than 10%) indicates that only 1000 data can not support the training of the network for recognizing natural images of 1000 categories, our method still achieve acceptable accuracies and significantly outperforms other comparison methods for both VGGNet-16 and ResNet-50. For example, with $2\times$ acceleration rate, our pruned VGGNet-16 can achieve top-5 accuracy 89.4% while other methods fail to have acceptable classification performance (less than 85.27%). Note that for limited data ($\leq 100k$), the results of layer-wise reconstruction methods (DCP and ours) refer to the accuracy without final fine-tuning; we empirically find that the final fine-tuning almost makes no difference when training data are quite limited, *i.e.*, only about 0.1% improvement.

By varying the number of training data from 0.1k to 100k, Table 3 shows that the accuracy increases accordingly, and when the data are limited, our method enjoys great superiority. Moreover, we accelerate the ResNet-18 1.5 × (the pruned network with only 1.2B Flops), and show the top-5 accuracies with various numbers of training images in Table 4 . Results show that although ResNet-18 is more difficult

[4] https://pytorch.org/docs/master/torchvision/models.html

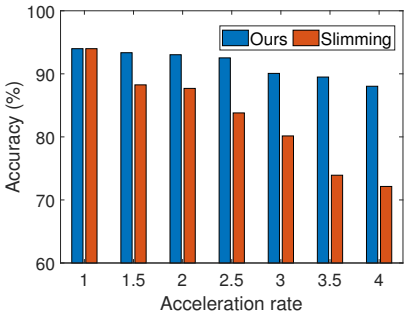


Figure 5: Accuracy drop of the pruned VGGNet on CIFAR-10 dataset with different acceleration rates (#FLOPs $\downarrow$). $N = 500$.

to prune than VGGNet-16, our method can still have satisfying classification performance with limited data, which validates the effectiveness of reborn filters to maintain the information of original filters. Note that for both VGGNet-16 and ResNet-18, when we use the entire training dataset, all comparison methods show comparable classification performance since it is sufficient for the pruned networks to learn useful filters from the data.

Ablation studies

Performance with different acceleration rates To study the performance limit of the pruned networks with different acceleration rates when the number of training data is fixed, we prune the VGGNet on the CIFAR-10 dataset with 1% training images. For fair comparison, the acceleration rate is controlled roughly from 1.5$\times$ to 4$\times$ by adjusting λ. When the acceleration rate is higher, it is more challenging to prune the networks with limited data since more channels are removed with significant information loss. As shown Figure 5, the accuracy of the pruned networks degrades with higher acceleration rate. However, compared with state-of-the-art Slimming(Liu et al. 2017), our method degrades much slower and the accuracy is still acceptable. For example, our method still achieves accuracy 90.07% with 3$\times$ acceleration rate while the accuracy of Slimming method is only 80.15%.

Effect of sparsity weight λ The sparsity weight λ controls sparse degree of the reborn filters as well as the pruning rate r. To understand the effect of λ, we focus on pruning a single layer (*i.e.*, `conv1_2`) of VGGNet-16 on ImageNet dataset following (Zhuang et al. 2018), as shown in Table 5. Large λ leads to prune more channels and get higher pruning rate r. Besides testing accuracy, we also report the reconstruction error of feature maps on the training data (5k images) and testing data (50k images), denoted as "ReErr-Train" and "ReErr-Test" in Table 5, respectively. For `conv1_2` in VGGNet-16, when λ is less than 0.3, the reconstruction error can almost be neglected (*i.e.*, smaller than 0.003), implying that our method squeezes the redundancy but preserves valuable information well. Note that the reconstruction error on test dataset is roughly equal to that on training dataset. It shows that our method generalizes well even when the training dataset is very limited since we inherit the original filters.

Table 5: The pruned VGGNet-16 on ImageNet dataset with different λ's for a single layer (*i.e.*, `conv1_2`). r refers to the pruning rate. ReErr-Train and ReErr-Test are the recontruction error on the training and testing dataset, respectively. Acc-Test is the testing top-5 accuracy. $N = 5k$.

λ	r	ReErr-Train	ReErr-Test	Acc-Test
0	0%	0	0	91.52
0.1	59%	0.0020	0.0021	91.45
0.2	73%	0.0032	0.0033	91.38
0.3	78%	0.0030	0.0030	91.40
0.4	81%	0.0072	0.0075	91.16
0.5	86%	0.0141	0.0145	85.45

Visualization of reborn filters and feature maps

For better intuitive understanding, we implement visualization about reborn filters as well as the original filters of a layer (*i.e.*, `conv2_1`) in VGGNet-16 on ImageNet dataset. To show the development of reborn filters more visually, we display the feature maps produced by original filters and reborn filters, respectively. And for clarity we decompose the convolution of a single output channel into two steps, *i.e.*, convolution channel-wisely and summing the results.

As Figure 4 (b) shows, some reborn filters are actually constructed by multiple original filters, with different and enhanced feature maps obtained accordingly. Moreover, some reborn filters in Figure 4 (d) are of small values, and thus are reckoned as redundant filters and need pruned. Note that their corresponding original filters are not necessarily redundant since they are involved to develop other useful reborn filters. We also notice that some of reborn filters as Figure 4 (c) just inherit the original filters, *i.e.*, nearly copying, we argue this results from that these original filters are essentially informative and important for the performance of networks. Thus we can see that our reborn filters are capable of identifying the important channels automatically and re-generating newly valuable channels. More visualization results refer to the supplementary materials.

Conclusion

This paper introduces reborn filters for channel pruning of convolutional neural networks with limited data. The reborn filters inherit the informative structure priors of the original filters, so that the performance of original network can be maintained to a large extent. Moreover, reborn filters can be easily implemented using 1×1 convolution with linear activations, making it friendly for various applications in end-to-end fashion. Our proposed pruning algorithm shows the effectiveness and superiority on extensive experimental results. With quite limited (*e.g.*, 1‰) examples, our pruned network is still comparable to those trained with complete dataset on the classification accuracy. As for future work, we will investigate nonlinear fashion (*e.g.*, using networks) to model the construction of reborn filters using original filters, which may be related to the tools in meta-learning and continual learning.

Acknowledgments

This work is supported by National Natural Science Foundation of China under Grant No. 61876007, 61872012 and Australian Research Council under Project DE-180101438.

References

Alvarez, J. M., and Salzmann, M. 2016. Learning the number of neurons in deep networks. In *Advances in Neural Information Processing Systems*, 2270–2278.

Alvarez, J. M., and Salzmann, M. 2017. Compression-aware training of deep networks. In *Advances in Neural Information Processing Systems*, 856–867.

Boyd, S.; Parikh, N.; Chu, E.; Peleato, B.; Eckstein, J.; et al. 2011. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends® in Machine learning* 3(1):1–122.

Bucak, S. S.; Jin, R.; and Jain, A. K. 2014. Multiple kernel learning for visual object recognition: A review. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 36(7):1354–1369.

Cheng, Y.; Wang, D.; Zhou, P.; and Zhang, T. 2017. A survey of model compression and acceleration for deep neural networks. *arXiv preprint arXiv:1710.09282*.

Courbariaux, M.; Bengio, Y.; and David, J.-P. 2015. Binaryconnect: Training deep neural networks with binary weights during propagations. In *Advances in neural information processing systems*, 3123–3131.

Gönen, M., and Alpaydın, E. 2011. Multiple kernel learning algorithms. *Journal of machine learning research* 12(Jul):2211–2268.

Grant, M., and Boyd, S. 2014. CVX: Matlab software for disciplined convex programming, version 2.1. http://cvxr.com/cvx.

Han, S.; Mao, H.; and Dally, W. J. 2015. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. *arXiv preprint arXiv:1510.00149*.

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 770–778.

He, Y.; Zhang, X.; and Sun, J. 2017. Channel pruning for accelerating very deep neural networks. In *Proceedings of the IEEE International Conference on Computer Vision*, 1389–1397.

Kim, J.; Park, S.; and Kwak, N. 2018. Paraphrasing complex network: Network compression via factor transfer. In Bengio, S.; Wallach, H.; Larochelle, H.; Grauman, K.; Cesa-Bianchi, N.; and Garnett, R., eds., *Advances in Neural Information Processing Systems 31*. Curran Associates, Inc. 2760–2769.

Lebedev, V.; Ganin, Y.; Rakhuba, M.; Oseledets, I.; and Lempitsky, V. 2014. Speeding-up convolutional neural networks using finetuned cp-decomposition. *arXiv preprint arXiv:1412.6553*.

Li, H.; De, S.; Xu, Z.; Studer, C.; Samet, H.; and Goldstein, T. 2017. Training quantized nets: A deeper understanding. In Guyon, I.; Luxburg, U. V.; Bengio, S.; Wallach, H.; Fergus, R.; Vishwanathan, S.; and Garnett, R., eds., *Advances in Neural Information Processing Systems 30*. Curran Associates, Inc. 5811–5821.

Li, T.; Li, J.; Liu, Z.; and Zhang, C. 2018. Knowledge distillation from few samples. *arXiv preprint arXiv:1812.01839*.

Liu, Z.; Li, J.; Shen, Z.; Huang, G.; Yan, S.; and Zhang, C. 2017. Learning efficient convolutional networks through network slimming. In *Proceedings of the IEEE International Conference on Computer Vision*, 2736–2744.

Liu, Z.; Sun, M.; Zhou, T.; Huang, G.; and Darrell, T. 2018. Rethinking the value of network pruning. *arXiv preprint arXiv:1810.05270*.

Louizos, C.; Ullrich, K.; and Welling, M. 2017. Bayesian compression for deep learning. In Guyon, I.; Luxburg, U. V.; Bengio, S.; Wallach, H.; Fergus, R.; Vishwanathan, S.; and Garnett, R., eds., *Advances in Neural Information Processing Systems 30*. Curran Associates, Inc. 3288–3298.

Luo, J.-H.; Wu, J.; and Lin, W. 2017. Thinet: A filter level pruning method for deep neural network compression. In *Proceedings of the IEEE international conference on computer vision*, 5058–5066.

Mahajan, D.; Girshick, R.; Ramanathan, V.; He, K.; Paluri, M.; Li, Y.; Bharambe, A.; and van der Maaten, L. 2018. Exploring the limits of weakly supervised pretraining. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 181–196.

Mairal, J.; Bach, F.; Ponce, J.; and Sapiro, G. 2010. Online learning for matrix factorization and sparse coding. *Journal of Machine Learning Research* 11(Jan):19–60.

Novikov, A.; Podoprikhin, D.; Osokin, A.; and Vetrov, D. P. 2015. Tensorizing neural networks. In Cortes, C.; Lawrence, N. D.; Lee, D. D.; Sugiyama, M.; and Garnett, R., eds., *Advances in Neural Information Processing Systems 28*. Curran Associates, Inc. 442–450.

Paszke, A.; Gross, S.; Chintala, S.; Chanan, G.; Yang, E.; DeVito, Z.; Lin, Z.; Desmaison, A.; Antiga, L.; and Lerer, A. 2017. Automatic differentiation in pytorch.

Rastegari, M.; Ordonez, V.; Redmon, J.; and Farhadi, A. 2016. Xnor-net: Imagenet classification using binary convolutional neural networks. In *European Conference on Computer Vision*, 525–542. Springer.

Russakovsky, O.; Deng, J.; Su, H.; Krause, J.; Satheesh, S.; Ma, S.; Huang, Z.; Karpathy, A.; Khosla, A.; Bernstein, M.; Berg, A. C.; and Fei-Fei, L. 2015. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)* 115(3):211–252.

Simon, N.; Friedman, J.; Hastie, T.; and Tibshirani, R. 2013. A sparse-group lasso. *Journal of Computational and Graphical Statistics* 22(2):231–245.

Simonyan, K., and Zisserman, A. 2014. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*.

Wang, Y.; Xu, C.; You, S.; Tao, D.; and Xu, C. 2016. Cnnpack: Packing convolutional neural networks in the frequency domain. In Lee, D. D.; Sugiyama, M.; Luxburg, U. V.; Guyon, I.; and Garnett, R., eds., *Advances in Neural Information Processing Systems 29*. Curran Associates, Inc. 253–261.

Wang, C.; Li, M.; and Smola, A. J. 2019. Language models with transformers. *CoRR* abs/1904.09408.

Wen, W.; Wu, C.; Wang, Y.; Chen, Y.; and Li, H. 2016. Learning structured sparsity in deep neural networks. In Lee, D. D.; Sugiyama, M.; Luxburg, U. V.; Guyon, I.; and Garnett, R., eds., *Advances in Neural Information Processing Systems 29*. Curran Associates, Inc. 2074–2082.

You, S.; Xu, C.; Wang, Y.; Xu, C.; and Tao, D. 2017a. Privileged multi-label learning. *arXiv preprint arXiv:1701.07194*.

You, S.; Xu, C.; Xu, C.; and Tao, D. 2017b. Learning from multiple teacher networks. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 1285–1294. ACM.

Zhong, W., and Kwok, J. 2014. Fast stochastic alternating direction method of multipliers. In *International Conference on Machine Learning*, 46–54.

Zhuang, Z.; Tan, M.; Zhuang, B.; Liu, J.; Guo, Y.; Wu, Q.; Huang, J.; and Zhu, J. 2018. Discrimination-aware channel pruning for deep neural networks. In *Advances in Neural Information Processing Systems*, 883–894.