Soft Filter Pruning for Accelerating Deep Convolutional Neural Networks

Yang He[1,2], **Guoliang Kang**[2], **Xuanyi Dong**[2], **Yanwei Fu**[3]*, **Yi Yang**[1,2]*
[1]SUSTech-UTS Joint Centre of CIS, Southern University of Science and Technology
[2]CAI, University of Technology Sydney
[3]The School of Data Science, Fudan University
{yang.he-1, guoliang.kang, xuanyi.dong}@student.uts.edu.au,
yanweifu@fudan.edu.cn, yi.yang@uts.edu.au

Abstract

This paper proposed a Soft Filter Pruning (SFP) method to accelerate the inference procedure of deep Convolutional Neural Networks (CNNs). Specifically, the proposed SFP enables the pruned filters to be updated when training the model after pruning. SFP has two advantages over previous works: (1) **Larger model capacity.** Updating previously pruned filters provides our approach with larger optimization space than fixing the filters to zero. Therefore, the network trained by our method has a larger model capacity to learn from the training data. (2) **Less dependence on the pre-trained model.** Large capacity enables SFP to train from scratch and prune the model simultaneously. In contrast, previous filter pruning methods should be conducted on the basis of the pre-trained model to guarantee their performance. Empirically, SFP from scratch outperforms the previous filter pruning methods. Moreover, our approach has been demonstrated effective for many advanced CNN architectures. Notably, on ILSCRC-2012, SFP reduces more than 42% FLOPs on ResNet-101 with even 0.2% top-5 accuracy improvement, which has advanced the state-of-the-art. Code is publicly available on GitHub: https://github.com/he-y/soft-filter-pruning

1 Introduction

The superior performance of deep CNNs usually comes from the deeper and wider architectures, which cause the prohibitively expensive computation cost. Even if we use more efficient architectures, such as residual connection [He *et al.*, 2016a] or inception module [Szegedy *et al.*, 2015], it is still difficult in deploying the state-of-the-art CNN models on mobile devices. For example, ResNet-152 has 60.2 million parameters with 231MB storage spaces; besides, it also needs more than 380MB memory footprint and six seconds (11.3 billion float point operations, FLOPs) to process a single image on CPU. The storage, memory, and computation of this cumbersome model significantly exceed the computing limitation of current mobile devices. Therefore, it is essential to maintain the small size of the deep CNN models which has relatively low computational cost but high accuracy in real-world applications.

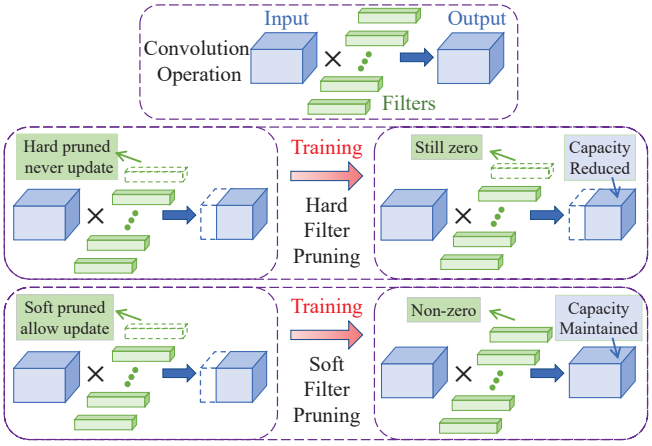


Figure 1: **Hard Filter Pruning** *v.s.* **Soft Filter Pruning**. We mark the pruned filter as the green dashed box. For the hard filter pruning, the pruned filters are always ***fixed*** during the whole training procedure. Therefore, the model capacity is reduced and thus harms the performance because the dashed blue box is useless during training. On the contrary, our SFP ***allows*** the pruned filters to be updated during the training procedure. In this way, the model capacity is recovered from the pruned model, and thus leads a better accuracy.

Recent efforts have been made either on directly deleting weight values of filters [Han *et al.*, 2015b] (i.e., weight pruning) or totally discarding some filters (i.e., filter pruning) [Li *et al.*, 2017; He *et al.*, 2017; Luo *et al.*, 2017]. However, the weight pruning may result in the unstructured sparsity of filters, which may still be less efficient in saving the memory usage and computational cost, since the unstructured model cannot leverage the existing high-efficiency BLAS libraries. In contrast, the filter pruning enables the model with structured sparsity and more efficient memory usage than weight pruning, and thus takes full advantage of BLAS libraries to achieve a more realistic acceleration. Therefore, the filter pruning is more advocated in accelerating the networks.

Nevertheless, most of the previous works on filter pruning

*Corrsponding Author

still suffer from the problems of (1) *the model capacity reduction* and (2) *the dependence on pre-trained model*. Specifically, as shown in Fig. 1, most previous works conduct the "hard filter pruning", which directly delete the pruned filters. The discarded filters will reduce the model capacity of original models, and thus inevitably harm the performance. Moreover, to maintain a reasonable performance with respect to the full models, previous works [Li *et al.*, 2017; He *et al.*, 2017; Luo *et al.*, 2017] always fine-tuned the hard pruned model after pruning the filters of a pre-trained model, which however has low training efficiency and often requires much more training time than the traditional training schema.

To address the above mentioned two problems, we propose a novel Soft Filter Pruning (SFP) approach. The SFP dynamically prunes the filters in a soft manner. Particularly, before first training epoch, the filters of almost all layers with small ℓ_2-norm are selected and set to zero. Then the training data is used to update the pruned model. Before the next training epoch, our SFP will prune a new set of filters of small ℓ_2-norm. These training process is continued until converged. Finally, some filters will be selected and pruned without further updating. The SFP algorithm enables the compressed network to have a larger model capacity, and thus achieve a higher accuracy than others.

Contributions. We highlight three contributions: (1) We propose SFP to allow the pruned filters to be updated during the training procedure. This soft manner can dramatically maintain the model capacity and thus achieves the superior performance. (2) Our acceleration approach can train a model from scratch and achieve better performance compared to the state-of-the-art. In this way, the fine-tuning procedure and the overall training time is saved. Moreover, using the pre-trained model can further enhance the performance of our approach to advance the state-of-the-art in model acceleration. (3) The extensive experiment on two benchmark datasets demonstrates the effectiveness and efficiency of our SFP. We accelerate ResNet-110 by two times with about 4% relative accuracy improvement on CIFAR-10, and also achieve state-of-the-art results on ILSVRC-2012.

2 Related Works

Most previous works on accelerating CNNs can be roughly divided into three categories, namely, *matrix decomposition*, *low-precision weights*, and *pruning*. In particular, the *matrix decomposition* of deep CNN tensors is approximated by the product of two low-rank matrices [Jaderberg *et al.*, 2014; Zhang *et al.*, 2016; Tai *et al.*, 2016]. This can save the computational cost. Some works [Zhu *et al.*, 2017; Zhou *et al.*, 2017] focus on compressing the CNNs by using *low-precision weights*. *Pruning*-based approaches aim to remove the unnecessary connections of the neural network [Han *et al.*, 2015b; Li *et al.*, 2017]. Essentially, the work of this paper is based on the idea of pruning techniques; and the approaches of matrix decomposition and low-precision weights are orthogonal but potentially useful here – it may be still worth simplifying the weight matrix after pruning filters, which would be taken as future work.

Weight Pruning. Many recent works [Han *et al.*, 2015b; 2015a; Guo *et al.*, 2016] pruning weights of neural network resulting in small models. For example, [Han *et al.*, 2015b] proposed an iterative weight pruning method by discarding the small weights whose values are below the threshold. [Guo *et al.*, 2016] proposed the dynamic network surgery to reduce the training iteration while maintaining a good prediction accuracy. [Wen *et al.*, 2016; Lebedev and Lempitsky, 2016] leveraged the sparsity property of feature maps or weight parameters to accelerate the CNN models. A special case of weight pruning is neuron pruning. However, pruning weights always leads to unstructured models, so the model cannot leverage the existing efficient BLAS libraries in practice. Therefore, it is difficult for weight pruning to achieve realistic speedup.

Filter Pruning. Concurrently with our work, some filter pruning strategies [Li *et al.*, 2017; Liu *et al.*, 2017; He *et al.*, 2017; Luo *et al.*, 2017] have been explored. Pruning the filters leads to the removal of the corresponding feature maps. This not only reduces the storage usage on devices but also decreases the memory footprint consumption to accelerate the inference. [Li *et al.*, 2017] uses ℓ_1-norm to select unimportant filters and explores the sensitivity of layers for filter pruning. [Liu *et al.*, 2017] introduces ℓ_1 regularization on the scaling factors in batch normalization (BN) layers as a penalty term, and prune channel with small scaling factors in BN layers. [Molchanov *et al.*, 2017] proposes a Taylor expansion based pruning criterion to approximate the change in the cost function induced by pruning. [Luo *et al.*, 2017] adopts the statistics information from next layer to guide the importance evaluation of filters. [He *et al.*, 2017] proposes a LASSO-based channel selection strategy, and a least square reconstruction algorithm to prune filers. However, for all these filter pruning methods, the representative capacity of neural network after pruning is seriously affected by smaller optimization space.

Discussion. To the best of our knowledge, there is only one approach that uses the soft manner to prune weights [Guo *et al.*, 2016]. We would like to highlight our advantages compared to this approach as below: (1) Our SPF focuses on the filter pruning, but they focus on the weight pruning. As discussed above, weight pruning approaches lack the practical implementations to achieve the realistic acceleration. (2) [Guo *et al.*, 2016] paid more attention to the model compression, whereas our approach can achieve both compression and acceleration of the model. (3) Extensive experiments have been conducted to validate the effectiveness of our proposed approach both on large-scale datasets and the state-of-the-art CNN models. In contrast, [Guo *et al.*, 2016] only had the experiments on Alexnet which is more redundant the advanced models, such as ResNet.

3 Methodology

3.1 Preliminaries

We will formally introduce the symbol and annotations in this section. The deep CNN network can be parameterized by $\{\mathbf{W}^{(i)} \in \mathbb{R}^{N_{i+1}\times N_i\times K\times K}, 1 \le i \le L\}$ $\mathbf{W}^{(i)}$ denotes a matrix of connection weights in the i-th layer. N_i denotes the number of input channels for the i-th convolution layer. L

denotes the number of layers. The shapes of input tensor $\mathbf{U}$ and output tensor $\mathbf{V}$ are $N_i \times H_i \times W_i$ and $N_{i+1} \times H_{i+1} \times W_{i+1}$, respectively. The convolutional operation of the i-th layer can be written as:

$$\mathbf{V}_{i,j} = \mathcal{F}_{i,j} * \mathbf{U} \text{ for } 1 \leq j \leq N_{i+1}, \tag{1}$$

where $\mathcal{F}_{i,j} \in \mathbb{R}^{N_i \times K \times K}$ represents the j-th filter of the i-th layer. $\mathbf{W}^{(i)}$ consists of $\{\mathcal{F}_{i,j}, 1 \leq j \leq N_{i+1}\}$. The $\mathbf{V}_{i,j}$ represents the j-th output feature map of the i-th layer.

Pruning filters can remove the output feature maps. In this way, the computational cost of the neural network will reduce remarkably. Let us assume the pruning rate of SFP is P_i for the i-th layer. The number of filters of this layer will be reduced from N_{i+1} to $N_{i+1}(1 - P_i)$, thereby the size of the output tensor $\mathbf{V}_{i,j}$ can be reduced to $N_{i+1}(1 - P_i) \times H_{i+1} \times W_{i+1}$. As the output tensor of i-th layer is the input tensor of $i + 1$-th layer, we can reduce the input size of i-th layer to achieve a higher acceleration ratio.

3.2 Soft Filter Pruning (SFP)

Most of previous filter pruning works [Li *et al.*, 2017; Liu *et al.*, 2017; He *et al.*, 2017; Luo *et al.*, 2017] compressed the deep CNNs in a hard manner. We call them as the hard filter pruning. Typically, these algorithms firstly prune filters of a single layer of a pre-trained model and fine-tune the pruned model to complement the degrade of the performance. Then they prune the next layer and fine-tune the model again until the last layer of the model is pruned. However, once filters are pruned, these approaches will not update these filters again. Therefore, the model capacity is drastically reduced due to the removed filters; and such a hard pruning manner affects the performance of the compressed models negatively.

As summarized in Alg. 1, the proposed SFP algorithm can dynamically remove the filters in a soft manner. Specifically, the key is to keep updating the pruned filters in the training stage. Such an updating manner brings several benefits. It not only keeps the model capacity of the compressed deep CNN models as the original models, but also avoids the greedy layer by layer pruning procedure and enable pruning almost *all layers* at the same time. More specifically, our approach can prune a model either in the process of training from scratch, or a pre-trained model. In each training epoch, the full model is optimized and trained on the training data. After each epoch, the ℓ_2-norm of all filters are computed for each weighted layer and used as the criterion of our filter selection strategy. Then we will prune the selected filters by setting the corresponding filter weights as zero, which is followed by next training epoch. Finally, the original deep CNNs are pruned into a compact and efficient model. The details of SFP is illustratively explained in Alg. 1, which can be divided into the following four steps.

Filter selection. We use the ℓ_p-norm to evaluate the importance of each filter as Eq. (2). In general, the convolutional results of the filter with the smaller ℓ_p-norm lead to relatively lower activation values; and thus have a less numerical impact on the final prediction of deep CNN models. In term of this understanding, such filters of small ℓ_p-norm will be given high priority of being pruned than those of higher ℓ_p-norm. Particularly, we use a pruning rate P_i to select $N_{i+1}P_i$ unimportant filters for the i-th weighted layer. In other words, the lowest $N_{i+1}P_i$ filters are selected, e.g., the blue filters in Fig. 2. In practice, ℓ_2-norm is used based on the empirical analysis.

Algorithm 1 Algorithm Description of SFP

Input: training data: $\mathbf{X}$, pruning rate: P_i
the model with parameters $\mathbf{W} = \{\mathbf{W}^{(i)}, 0 \leq i \leq L\}$.
Initialize the model parameter $\mathbf{W}$
for $epoch = 1$; $epoch \leq epoch_{max}$; $epoch++$ **do**
  Update the model parameter $\mathbf{W}$ based on $\mathbf{X}$
  for $i = 1$; $i \leq L$; $i++$ **do**
    Calculate the ℓ_2-norm for each filter $\|\mathcal{F}_{i,j}\|_2, 1 \leq j \leq N_{i+1}$
    Zeroize $N_{i+1}P_i$ filters by ℓ_2-norm filter selection
  end for
end for
Obtain the compact model with parameters $\mathbf{W}^*$ from $\mathbf{W}$
Output: The compact model and its parameters $\mathbf{W}^*$

$$\|\mathcal{F}_{i,j}\|_p = \sqrt[p]{\sum_{n=1}^{N_i} \sum_{k_1=1}^{K} \sum_{k_2=1}^{K} |\mathcal{F}_{i,j}(n, k_1, k_2)|^p}, \tag{2}$$

Filter Pruning. We set the value of selected $N_{i+1}P_i$ filters to zero (see the filter pruning step in Fig. 2). This can temporarily eliminate their contribution to the network output. Nevertheless, in the following training stage, we still allow these selected filters to be updated, in order to keep the representative capacity and the high performance of the model.

In the filter pruning step, we simply prune *all* the weighted layers at the same time. In this way, we can prune each filter in parallel, which would cost negligible computation time. In contrast, the previous filter pruning methods always conduct layer by layer greedy pruning. After pruning filters of one single layer, existing methods always require training to converge the network [Luo *et al.*, 2017; He *et al.*, 2017]. This procedure cost much extra computation time, especially when the depth increases. Moreover, we use the *same* pruning rate for *all* weighted layers. Therefore, we need only one hyper-parameter $P_i = P$ to balance the acceleration and accuracy. This can avoid the inconvenient hyper-parameter search or the complicated sensitivity analysis [Li *et al.*, 2017]. As we allow the pruned filters to be updated, the model has a large model capacity and becomes more flexible and thus can well balance the contribution of each filter to the final prediction.

Reconstruction. After the pruning step, we train the network for one epoch to reconstruct the pruned filters. As shown in Fig. 2, the pruned filters are updated to non-zero by back-propagation. In this way, SFP allows the pruned model to have the same capacity as the original model during training. In contrast, hard filter pruning decreases the number of feature maps. The reduction of feature maps would dramatically reduce the model capacity, and further harm the performance. Previous pruning methods usually require a pre-trained model and then fine-tune it. However, as we integrate the pruning step into the normal training schema, our approach can train the model from scratch. Therefore, the

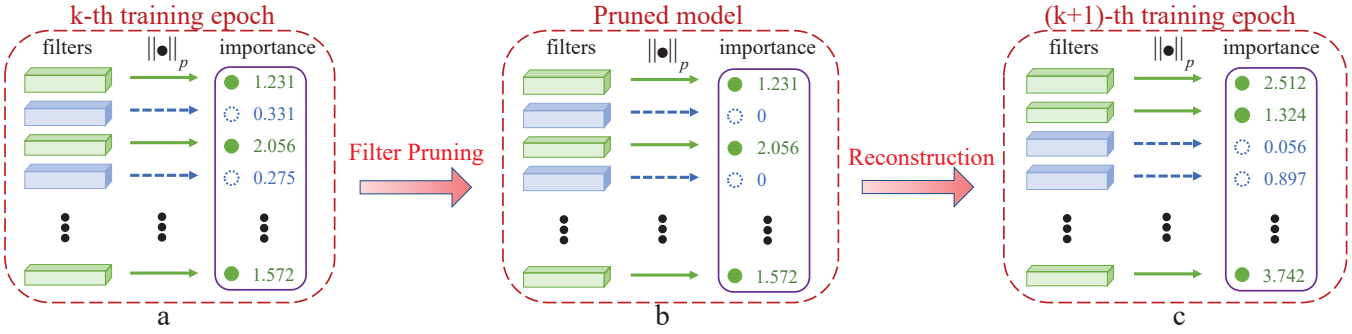


Figure 2: Overview of SFP. At the end of each training epoch, we prune the filters based on their importance evaluations. The filters are ranked by their ℓ_p-norms (purple rectangles) and the small ones (blue circles) are selected to be pruned. After filter pruning, the model undergoes a reconstruction process where pruned filters are capable of being reconstructed (i.e., updated from zeros) by the forward-backward process. **(a)**: filter instantiations before pruning. **(b)**: filter instantiations after pruning. **(c)**: filter instantiations after reconstruction.

fine-tuning stage is no longer necessary for SFP. As we will show in experiments, the network trained from scratch by SFP can obtain the competitive results with the one trained from a well-trained model by others. By leveraging the pre-trained model, SFP obtains a much higher performance and advances the state-of-the-art.

Obtaining Compact Model. SFP iterates over the filter selection, filter pruning and reconstruction steps. After the model gets converged, we can obtain a sparse model containing many "zero filters". One "zero filter" corresponds to one feature map. The features maps, corresponding to those "zero filters", will always be zero during the inference procedure. There will be no influence to remove these filters as well as the corresponding feature maps. Specifically, for the pruning rate P_i in the i-th layer, only $N_{i+1}(1-P_i)$ filters are non-zero and have an effect on the final prediction. Consider pruning the previous layer, the input channel of i-th layer is changed from N_i to $N_i(1-P_{i-1})$. We can thus re-build the i-th layer into a smaller one. Finally, a compact model $\{\mathbf{W}^{*(i)} \in \mathbb{R}^{N_{i+1}(1-P_i)\times N_i(1-P_{i-1})\times K\times K}\}$ is obtained.

3.3 Computation Complexity Analysis

Theoretical speedup analysis. Suppose the filter pruning rate of the ith layer is P_i, which means the $N_{i+1} \times P_i$ filters are set to zero and pruned from the layer, and the other $N_{i+1} \times (1-P_i)$ filters remain unchanged, and suppose the size of the input and output feature map of ith layer is $H_i \times W_i$ and $H_{i+1} \times W_{i+1}$. Then after filter pruning, the dimension of useful output feature map of the ith layer decreases from $N_{i+1} \times H_{i+1} \times W_{i+1}$ to $N_{i+1}(1-P_i) \times H_{i+1} \times W_{i+1}$. Note that the output of ith layer is the input of $(i+1)$ th layer. And we further prunes the $(i+1)$th layer with a filter pruning rate P_{i+1}, then the calculation of $(i+1)$th layer is decrease from $N_{i+2} \times N_{i+1} \times k^2 \times H_{i+2} \times W_{i+2}$ to $N_{i+2}(1-P_{i+1}) \times N_{i+1}(1-P_i) \times k^2 \times H_{i+2} \times W_{i+2}$. In other words, a proportion of $1-(1-P_{i+1}) \times (1-P_i)$ of the original calculation is reduced, which will make the neural network inference much faster.

Realistic speedup analysis. In theoretical speedup analysis, other operations such as batch normalization (BN) and pooling are negligible comparing to convolution operations. Therefore, we consider the FLOPs of convolution operations for computation complexity comparison, which is commonly used in previous work [Li *et al.*, 2017; Luo *et al.*, 2017]. However, reduced FLOPs cannot bring the same level of realistic speedup because non-tensor layers (e.g., BN and pooling layers) also need the inference time on GPU [Luo *et al.*, 2017]. In addition, the limitation of IO delay, buffer switch and efficiency of BLAS libraries also lead to the wide gap between theoretical and realistic speedup ratio. We compare the theoretical and realistic speedup in Section 4.3.

4 Evaluation and Results

4.1 Benchmark Datasets and Experimental Setting

Our method is evaluated on two benchmarks: CIFAR-10 [Krizhevsky and Hinton, 2009] and ILSVRC-2012 [Russakovsky *et al.*, 2015]. The CIFAR-10 dataset contains 50,000 training images and 10,000 testing images, which are categorized into 10 classes. ILSVRC-2012 is a large-scale dataset containing 1.28 million training images and 50k validation images of 1,000 classes. Following the common setting in [Luo *et al.*, 2017; He *et al.*, 2017; Dong *et al.*, 2017a], we focus on pruning the challenging ResNet model in this paper. SFP should also be effective on different computer vision tasks, such as [Kang *et al.*, 2017; Ren *et al.*, 2015; Dong *et al.*, 2018; Shen *et al.*, 2018b; Yang *et al.*, 2010; Shen *et al.*, 2018a; Dong *et al.*, 2017b], and we will explore this in future.

In the CIFAR-10 experiments, we use the default parameter setting as [He *et al.*, 2016b] and follow the training schedule in [Zagoruyko and Komodakis, 2016]. On ILSVRC-2012, we follow the same parameter settings as [He *et al.*, 2016a; 2016b]. We use the same data argumentation strategies with PyTorch official examples [Paszke *et al.*, 2017].

We conduct our SFP operation at the end of every training epoch. For pruning a scratch model, we use the normal training schedule. For pruning a pre-trained model, we reduce the learning rate by 10 compared to the schedule for the scratch model. We run each experiment three times and report the "mean $\pm$ std". We compare the performance with other state-of-the-art acceleration algorithms, e.g., [Dong *et al.*, 2017a; Li *et al.*, 2017; He *et al.*, 2017; Luo *et al.*, 2017].

Depth	Method	Fine-tune?	Baseline Accu. (%)	Accelerated Accu. (%)	Accu. Drop (%)	FLOPs	Pruned FLOPs(%)
20	[Dong *et al.*, 2017a]	N	91.53	91.43	0.10	3.20E7	20.3
	Ours(10%)	N	**92.20 ± 0.18**	**92.24 ± 0.33**	**-0.04**	3.44E7	15.2
	Ours(20%)	N	**92.20 ± 0.18**	91.20 ± 0.30	1.00	2.87E7	29.3
	Ours(30%)	N	**92.20 ± 0.18**	90.83 ± 0.31	1.37	**2.43E7**	**42.2**
32	[Dong *et al.*, 2017a]	N	92.33	90.74	1.59	4.70E7	31.2
	Ours(10%)	N	**92.63 ± 0.70**	**93.22 ± 0.09**	**-0.59**	5.86E7	14.9
	Ours(20%)	N	**92.63 ± 0.70**	90.63 ± 0.37	0.00	4.90E7	28.8
	Ours(30%)	N	**92.63 ± 0.70**	90.08 ± 0.08	0.55	**4.03E7**	**41.5**
56	[Li *et al.*, 2017]	N	93.04	91.31	1.75	9.09E7	27.6
	[Li *et al.*, 2017]	Y	93.04	93.06	-0.02	9.09E7	27.6
	[He *et al.*, 2017]	N	92.80	90.90	1.90	-	50.0
	[He *et al.*, 2017]	Y	92.80	91.80	1.00	-	50.0
	Ours(10%)	N	**93.59 ± 0.58**	**93.89 ± 0.19**	**-0.30**	1.070E8	14.7
	Ours(20%)	N	**93.59 ± 0.58**	93.47 ± 0.24	0.12	8.98E7	28.4
	Ours(30%)	N	**93.59 ± 0.58**	93.10 ± 0.20	0.49	7.40E7	41.1
	Ours(30%)	Y	**93.59 ± 0.58**	93.78 ± 0.22	-0.19	7.40E7	41.1
	Ours(40%)	N	**93.59 ± 0.58**	92.26 ± 0.31	1.33	**5.94E7**	**52.6**
	Ours(40%)	Y	**93.59 ± 0.58**	93.35 ± 0.31	0.24	**5.94E7**	**52.6**
110	[Li *et al.*, 2017]	N	93.53	92.94	0.61	1.55E8	38.6
	[Li *et al.*, 2017]	Y	93.53	93.30	0.20	1.55E8	38.6
	[Dong *et al.*, 2017a]	N	93.63	93.44	0.19	-	34.2
	Ours(10%)	N	**93.68 ± 0.32**	93.83 ± 0.19	-0.15	2.16E8	14.6
	Ours(20%)	N	**93.68 ± 0.32**	**93.93 ± 0.41**	**-0.25**	1.82E8	28.2
	Ours(30%)	N	**93.68 ± 0.32**	93.38 ± 0.30	0.30	**1.50E8**	**40.8**
	Ours(30%)	Y	**93.68 ± 0.32**	93.86 ± 0.21	-0.18	**1.50E8**	**40.8**

Table 1: Comparison of pruning ResNet on CIFAR-10. In "Fine-tune?" column, "Y" and "N" indicate whether to use the pre-trained model as initialization or not, respectively. The "Accu. Drop" is the accuracy of the pruned model minus that of the baseline model, so negative number means the accelerated model has a higher accuracy than the baseline model. A smaller number of "Accu. Drop" is better.

4.2 ResNet on CIFAR-10

Settings. For CIFAR-10 dataset, we test our SFP on ResNet-20, 32, 56 and 110. We use several different pruning rates, and also analyze the difference between using the pre-trained model and from scratch.

Results. Tab. 1 shows the results. Our SFP could achieve a better performance than the other state-of-the-art hard filter pruning methods. For example, [Li *et al.*, 2017] use the hard pruning method to accelerate ResNet-110 by 38.6% speedup ratio with 0.61% accuracy drop when without fine-tuning. When using pre-trained model and fine-tuning, the accuracy drop becomes 0.20%. However, we can accelerate the inference of ResNet-110 to 40.8% speed-up with only 0.30% accuracy drop without fine-tuning. When using the pre-trained model, we can even outperform the original model by 0.18% with about more than 40% FLOPs reduced.

These results validate the effectiveness of SFP, which can produce a more compressed model with comparable performance to the original model.

4.3 ResNet on ILSVRC-2012

Settings. For ILSVRC-2012 dataset, we test our SFP on ResNet-18, 34, 50 and 101; and we use the same pruning rate 30% for all the models. All the convolutional layer of ResNet are pruned with the same pruning rate at the same time. (We do not prune the projection shortcuts for simplification, which only need negligible time and do not affect the overall cost.)

Results. Tab. 2 shows that SFP outperforms other state-of-the-art methods. For ResNet-34, SFP without fine-tuning achieves more inference speedup to the hard pruning method [Luo *et al.*, 2017], but the accuracy of our pruned model exceeds their model by 2.57%. Moreover, for pruning a pre-trained ResNet-101, SFP reduces more than 40% FLOPs of the model with even 0.2% top-5 accuracy increase, which is the state-of-the-art result. In contrast, the performance degradation is inevitable for hard filter pruning method. Maintained model capacity of SFP is the main reason for the superior performance. In addition, the non-greedy all-layer pruning method may have a better performance than the locally optimal solution obtained from previous greedy pruning method, which seems to be another reason. Occasionally, large performance degradation happens for the pre-trained model (e.g., 14.01% top-1 accuracy drop for ResNet-50). This will be explored in our future work.

To test the realistic speedup ratio, we measure the forward time of the pruned models on one GTX1080 GPU with a batch size of 64 (shown in Tab. 3). The gap between theoretical and realistic model may come from and the limitation of IO delay, buffer switch and efficiency of BLAS libraries.

4.4 Ablation Study

We conducted extensive ablation studies to further analyze each component of SFP.

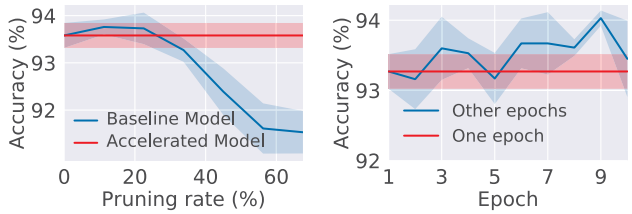
Filter Selection Criteria. The magnitude based criteria such as ℓ_p-norm are widely used to filter selection because computational resources cost is small [Li *et al.*, 2017]. We compare the ℓ_2-norm and ℓ_1-norm. For ℓ_1-norm criteria, the accuracy of the model under pruning rate 10%, 20%, 30% are 93.68±0.60%, 93.68±0.76% and 93.34±0.12%, respectively. While for ℓ_2-norm criteria, the accuracy

Depth	Method	Fine-tune?	Top-1 Accu. Baseline(%)	Top-1 Accu. Accelerated(%)	Top-5 Accu. Baseline(%)	Top-5 Accu. Accelerated(%)	Top-1 Accu. Drop(%)	Top-5 Accu. Drop(%)	Pruned FLOPs(%)
18	[Dong *et al.*, 2017a]	N	69.98	66.33	89.24	86.94	3.65	2.30	34.6
	Ours(30%)	N	**70.28**	**67.10**	**89.63**	**87.78**	**3.18**	**1.85**	**41.8**
34	[Dong *et al.*, 2017a]	N	73.42	**72.99**	91.36	**91.19**	**0.43**	**0.17**	24.8
	[Li *et al.*, 2017]	Y	73.23	72.17	-	-	1.06	-	24.2
	Ours(30%)	N	**73.92**	71.83	**91.62**	90.33	2.09	1.29	**41.1**
50	[He *et al.*, 2017]	Y	-	-	92.20	90.80	-	1.40	**50.0**
	[Luo *et al.*, 2017]	Y	72.88	72.04	91.14	90.67	**0.84**	**0.47**	36.7
	Ours(30%)	N	**76.15**	**74.61**	**92.87**	**92.06**	1.54	0.81	41.8
	Ours(30%)	Y	**76.15**	62.14	**92.87**	84.60	14.01	8.27	41.8
101	Ours(30%)	N	**77.37**	77.03	**93.56**	93.46	0.34	0.10	**42.2**
	Ours(30%)	Y	**77.37**	**77.51**	**93.56**	**93.71**	**-0.14**	**-0.20**	**42.2**

Table 2: Comparison of pruning ResNet on ImageNet. "Fine-tune?" and "Accu. Drop" have the same meaning with Tab. 1.

Model	Baseline time (ms)	Pruned time (ms)	Realistic Speed-up(%)	Theoretical Speed-up(%)
ResNet-18	37.10	26.97	27.4	41.8
ResNet-34	63.97	45.14	29.4	41.1
ResNet-50	135.01	94.66	29.8	41.8
ResNet-101	219.71	148.64	32.3	42.2

Table 3: Comparison on the theoretical and realistic speedup. We only count the time consumption of the forward procedure.



(a) Different Pruning Rates (b) Different SFP Intervals

Figure 3: Accuracy of ResNet-110 on CIFAR-10 regarding different hyper-parameters. (Solid line and shadow denotes the mean and standard deviation of three experiment, respectively.)

are 93.89±0.19%, 93.93±0.41% and 93.38±0.30%, respectively. The performance of ℓ_2-norm criteria is slightly better than that of ℓ_1-norm criteria. The result of ℓ_2-norm is dominated by the largest element, while the result of ℓ_1-norm is also largely affected by other small elements. Therefore, filters with some large weights would be preserved by the ℓ_2-norm criteria. So the corresponding discriminative features are kept so the performance of the pruned model is better.

Varying pruning rates. To comprehensively understand SFP, we test the accuracy of different pruning rates for ResNet-110, shown in Fig. 3(a). As the pruning rate increases, the accuracy of the pruned model first rises above the baseline model and then drops approximately linearly. For the pruning rate between 0% and about 23%, the accuracy of the accelerated model is higher than the baseline model. This shows that our SFP has a regularization effect on the neural network because SFP reduces the over-fitting of the model.

Sensitivity of SFP interval. By default, we conduct our SFP operation at the end of every training epoch. However, different SFP intervals may lead to different performance; so we explore the sensitivity of SFP interval. We use the ResNet-110 under pruning rate 30% as a baseline, and change the SFP interval from one epoch to ten epochs, as shown in Fig. 3(b). It is shown that the model accuracy has no large fluctuation along with the different SFP intervals. Moreover, the model accuracy of most (80%) intervals surpasses the accuracy of one epoch interval. Therefore, we can even achieve a better performance if we fine-tune this parameter.

Selection of pruned layers. Previous works always prune a portion of the layers of the network. Besides, different layers always have different pruning rates. For example, [Li *et al.*, 2017] only prunes insensitive layers, [Luo *et al.*, 2017] skips the last layer of every block of the ResNet, and [Luo *et al.*, 2017] prunes more aggressive for shallower layers and prune less for deep layers. Similarly, we compare the performance of pruning first and second layer of all basic blocks of ResNet-110. We set the pruning rate as 30%. The model with all the first layers of blocks pruned has an accuracy of $93.96 \pm 0.13\%$, while that with the second layers of blocks pruned has an accuracy of $93.38 \pm 0.44\%$. Therefore, different layers have different sensitivity for SFP, and careful selection of pruned layers would potentially lead to performance improvement, although more hyper-parameters are needed.

5 Conclusion and Future Work

In this paper, we propose a soft filter pruning (SFP) approach to accelerate the deep CNNs. During the training procedure, SFP allows the pruned filters to be updated. This soft manner can maintain the model capacity and thus achieve the superior performance. Remarkably, SFP can achieve the competitive performance compared to the state-of-the-art without the pre-trained model. Moreover, by leveraging the pre-trained model, SFP achieves a better result and advances the state-of-the-art. Furthermore, SFP can be combined with other acceleration algorithms, e.g., matrix decomposition and low-precision weights, to further improve the performance.

Acknowledgments

Yi Yang is the recipient of a Google Faculty Research Award. We acknowledge the Data to Decisions CRC (D2D CRC), the Cooperative Research Centres Programme and ARC's DECRA (project DE170101415) for funding this research. We thank Amazon for the AWS Cloud Credits.

References

[Dong *et al.*, 2017a] Xuanyi Dong, Junshi Huang, Yi Yang, and Shuicheng Yan. More is less: A more complicated network with less inference complexity. In *CVPR*, 2017.

[Dong *et al.*, 2017b] Xuanyi Dong, Deyu Meng, Fan Ma, and Yi Yang. A dual-network progressive approach to weakly supervised object detection. In *ACM Multimedia*, 2017.

[Dong *et al.*, 2018] Xuanyi Dong, Shoou-I Yu, Xinshuo Weng, Shih-En Wei, Yi Yang, and Yaser Sheikh. Supervision-by-Registration: An unsupervised approach to improve the precision of facial landmark detectors. In *CVPR*, 2018.

[Guo *et al.*, 2016] Yiwen Guo, Anbang Yao, and Yurong Chen. Dynamic network surgery for efficient DNNs. In *NIPS*, 2016.

[Han *et al.*, 2015a] Song Han, Huizi Mao, and William J Dally. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. In *ICLR*, 2015.

[Han *et al.*, 2015b] Song Han, Jeff Pool, John Tran, and William Dally. Learning both weights and connections for efficient neural network. In *NIPS*, 2015.

[He *et al.*, 2016a] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016.

[He *et al.*, 2016b] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Identity mappings in deep residual networks. In *ECCV*, 2016.

[He *et al.*, 2017] Yihui He, Xiangyu Zhang, and Jian Sun. Channel pruning for accelerating very deep neural networks. In *ICCV*, 2017.

[Jaderberg *et al.*, 2014] Max Jaderberg, Andrea Vedaldi, and Andrew Zisserman. Speeding up convolutional neural networks with low rank expansions. In *BMVC*, 2014.

[Kang *et al.*, 2017] Guoliang Kang, Jun Li, and Dacheng Tao. Shakeout: A new approach to regularized deep neural network training. *IEEE T-PAMI*, 2017.

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

[Lebedev and Lempitsky, 2016] Vadim Lebedev and Victor Lempitsky. Fast ConvNets using group-wise brain damage. In *CVPR*, 2016.

[Li *et al.*, 2017] Hao Li, Asim Kadav, Igor Durdanovic, Hanan Samet, and Hans Peter Graf. Pruning filters for efficient ConvNets. In *ICLR*, 2017.

[Liu *et al.*, 2017] Zhuang Liu, Jianguo Li, Zhiqiang Shen, Gao Huang, Shoumeng Yan, and Changshui Zhang. Learning efficient convolutional networks through network slimming. In *ICCV*, 2017.

[Luo *et al.*, 2017] Jian-Hao Luo, Jianxin Wu, and Weiyao Lin. ThiNet: A filter level pruning method for deep neural network compression. In *ICCV*, 2017.

[Molchanov *et al.*, 2017] Pavlo Molchanov, Stephen Tyree, Tero Karras, Timo Aila, and Jan Kautz. Pruning convolutional neural networks for resource efficient transfer learning. In *ICLR*, 2017.

[Paszke *et al.*, 2017] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. In *NIPS-W*, 2017.

[Ren *et al.*, 2015] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *NIPS*, 2015.

[Russakovsky *et al.*, 2015] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. ImageNet large scale visual recognition challenge. *IJCV*, 2015.

[Shen *et al.*, 2018a] Tao Shen, Tianyi Zhou, Guodong Long, Jing Jiang, Shirui Pan, and Chengqi Zhang. Disan: Directional self-attention network for rnn/cnn-free language understanding. In *AAAI*, 2018.

[Shen *et al.*, 2018b] Tao Shen, Tianyi Zhou, Guodong Long, Jing Jiang, and Chengqi Zhang. Bi-directional block self-attention for fast and memory-efficient sequence modeling. In *ICLR*, 2018.

[Szegedy *et al.*, 2015] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *CVPR*, 2015.

[Tai *et al.*, 2016] Cheng Tai, Tong Xiao, Yi Zhang, Xiaogang Wang, et al. Convolutional neural networks with low-rank regularization. In *ICLR*, 2016.

[Wen *et al.*, 2016] Wei Wen, Chunpeng Wu, Yandan Wang, Yiran Chen, and Hai Li. Learning structured sparsity in deep neural networks. In *NIPS*, 2016.

[Yang *et al.*, 2010] Yi Yang, Dong Xu, Feiping Nie, Shuicheng Yan, and Yueting Zhuang. Image clustering using local discriminant models and global integration. *IEEE T-IP*, 2010.

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

[Zhang *et al.*, 2016] Xiangyu Zhang, Jianhua Zou, Kaiming He, and Jian Sun. Accelerating very deep convolutional networks for classification and detection. *IEEE T-PAMI*, 2016.

[Zhou *et al.*, 2017] Aojun Zhou, Anbang Yao, Yiwen Guo, Lin Xu, and Yurong Chen. Incremental network quantization: Towards lossless cnns with low-precision weights. In *ICLR*, 2017.

[Zhu *et al.*, 2017] Chenzhuo Zhu, Song Han, Huizi Mao, and William J Dally. Trained ternary quantization. In *ICLR*, 2017.