Filter Sketch for Network Pruning

Mingbao Lin[1], **Rongrong Ji**[*1,3], **Shaojie Li**[1], **Qixiang Ye**[2,3], **Yonghong Tian**[4], **Jianzhuang Liu**[4], and **Qi Tian**[5]

[1]Media Analytics and Computing Laboratory, Department of Artificial Intelligence, School of Informatics, Xiamen University, China
[2]University of Chinese Academy of Sciences, Beijing, China
[3]Peng Cheng Laboratory, Shenzhen, China
[4]School of Electronics Engineering and Computer Science, Peking University, Beijing, China
[5]Huawei Noah's Ark Lab, China

Abstract

In this paper, we propose a novel network pruning approach by information preserving of pre-trained network weights (filters). Our approach, referred to as FilterSketch, encodes the second-order information of pre-trained weights, through which the model performance is recovered by fine-tuning the pruned network in an end-to-end manner. Network pruning with information preserving can be approximated as a matrix sketch problem, which is efficiently solved by the off-the-shelf Frequent Direction method. FilterSketch thereby requires neither training from scratch nor data-driven iterative optimization, leading to a magnitude-order reduction of time consumption in the optimization of pruning. Experiments on CIFAR-10 show that FilterSketch reduces 63.3% of FLOPs and prunes 59.9% of network parameters with negligible accuracy cost overhead for ResNet-110. On ILSVRC-2012, it achieves a reduction of 45.5% FLOPs and removes 43.0% of parameters with only a small top-5 accuracy drop of 0.69% for ResNet-50. Source codes of the proposed FilterSketch can be available at https://github.com/lmbxmu/FilterSketch.

1 Introduction

Deep convolutional neural networks (CNNs) are typically over-parameterized, which results in significant memory requirement and computational cost, hindering the deployment of CNN models on front-end systems of limited storage and computational power. Consequently, there is a growing need for reduction of model size by parameter quantization [Hubara *et al.*, 2016; Wang *et al.*, 2019], low-rank decomposition [Denil *et al.*, 2013; Hayashi *et al.*, 2019] and network pruning [Han *et al.*, 2015; Singh *et al.*, 2019]. Among them, network pruning emerges as a broad prospect in various applications. By removing redundant structures and/or parameters, network pruning achieves promising results in reducing CNNs' storage and/or computation cost.

*Corresponding author: rrji@xmu.edu.cn

Early works [LeCun *et al.*, 1990; Han *et al.*, 2015] about network pruning usually use unstructured methods to reduce network weights and obtain irregular sparsity of filters. Recent works pay more attention to structured pruning [Singh *et al.*, 2019; Zhao *et al.*, 2019; Lin *et al.*, 2019], which pursues not only reducing model size but also improving computational efficiency, facilitating CNNs' deployment on general-purpose hardware or high-efficient basic linear algebra subprograms (BLAS) libraries.

Existing structured pruning approaches can be classified into three categories: (1) Regularization-based pruning, which introduces hand-crafted regularizers, *e.g.*, sparse constraint [Liu *et al.*, 2017; Huang and Wang, 2018; Zhao *et al.*, 2019] and mask scheme [Lin *et al.*, 2019], into the training procedure. Despite of the simplicity, this kind of approaches usually requires to train from scratch and therefore is computationally expensive. (2) Property-based pruning. This line of works aims to pick up a specific property of a pre-trained network, *e.g.*, l_1-norm [Li *et al.*, 2017] and/or ratio of activations [Hu *et al.*, 2016], and filters with less importance are simply removed. However, many of these approaches require to prune the filters of a single pre-trained layer and fine-tune the pruned model. This layer-wise fine-tuning is recursively carried out until the last layer of the pre-trained model is pruned, which is very costly. (3) Optimization-based pruning. This kind of approaches converts pruning to a reconstruction problem and layer-wise optimization is employed to minimize the reconstruction error between the full model and the pruned model [He *et al.*, 2017; Luo *et al.*, 2017]. Nevertheless, the optimization formulations are typically intractable. To solve them, the constraints have to be relaxed, which inevitably weakens the performance. Moreover, the optimization procedure in each layer is typically data-driven and/or iterative [He *et al.*, 2017; Luo *et al.*, 2017], which inevitably brings a heavy optimization burden.

In this paper, we propose a novel approach termed FilterSketch, which, by encoding the second-order information of pre-trained network weights, provides a new perspective for deep CNN compression. FilterSketch is inspired by the fact that preserving the second-order covariance of a matrix

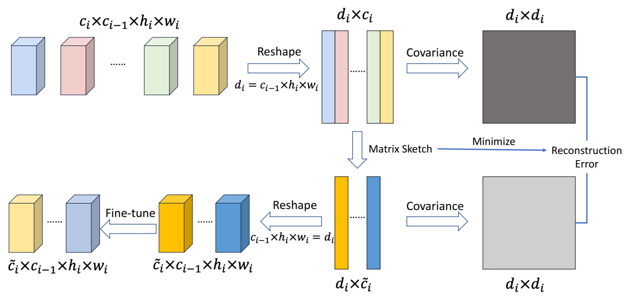


Figure 1: Framework of FilterSketch. The top displays the second-order covariance of the pre-trained CNN. The bottom shows the estimation of the second-order covariance for the pruned CNN. Our work preserves the covariance information in the pruned model, which is then effectively and efficiently solved by matrix sketch.

is equal to maximizing the correlation of multi-variate data [Sun *et al.*, 2017]. Such representations have been demonstrated to be effective in many other tasks [Kim *et al.*, 2018; Zhang *et al.*, 2019; Ren *et al.*, 2019] and provide the possibility for matrix compression, *e.g.*, network pruning. This possibility is proved to be true as shown in Sec. 4. Compared with existing pruning approaches, FilterSketch especially advantages in two aspects: (1) The learning procedure is comprehensively simplified as neither training-from-scratch nor layer-wise fine-tuning is required. (2) It has superiority of reducing network complexity over existing SOTAs.

To that effect, instead of simply discarding the unimportant filters, we propose to preserve the second-order information of the pre-trained model in the pruned model as shown in Fig. 1. For each layer in the pre-trained model, FilterSketch learns a set of new parameters for the pruned model which maintains the second-order covariance of the model. The new group of sketched parameters then serve as a warm-up for fine-tuning the pruned network. Though the fine-tuning is not layer-wise, the warm-up provides an excellent ability to recover model performance.

We show that preserving the second-order information can be approximated as a matrix sketch problem, which can then be efficiently solved by the off-the-shelf Frequent Direction method [Liberty, 2013], leading to a magnitude-order reduction of optimization time. FilterSketch thus involves neither complex regularization to restart a costly retraining and hyper-parameter tuning nor data-driven and/or iterative optimization to approximate the covariance information of the pre-trained model.

We evaluate the proposed FilterSketch by using popular CNNs with compact-designed components including GoogLeNet [Szegedy *et al.*, 2015] and ResNet [He *et al.*, 2016] with different network layers (56/110) on CIFAR-10 [Krizhevsky *et al.*, 2009] and ILSVRC-2012 [Russakovsky *et al.*, 2015]. Comparisons with state-of-the-art structured pruning approaches demonstrate the advantages of our approach in improving pruning efficiency and reducing the complexity of deep CNNs.

2 Related Work

Unstructured pruning and structured pruning are two major lines of methods for network model compression. In a broader view, *e.g.*, parameter quantization and low-rank decomposition can be integrated into network pruning to achieve higher compression and speedup.

Unstructured Pruning. Along with the increase of network complexity and model size, network pruning has attracted increasing attention. As a pioneer work, [LeCun *et al.*, 1990] utilized the second-order Taylor expansion to select less important parameters for deletion. [Han *et al.*, 2015] introduced an iterative weight pruning method by fine-tuning with a strong l_2 regularization and discarding the small weights with values below a threshold. Group sparsity based regularization of network parameters [Alvarez and Salzmann, 2016] was leveraged to penalize unimportant parameters. [Dong *et al.*, 2017] proposed to prune parameters based on the second-order derivatives of a layer-wise error function. [Liu *et al.*, 2018] implemented CNNs in the frequency domain and applied 2-D DCT transformation to sparsify the coefficients for spatial redundancy removal. After training a network, the lottery ticket hypothesis [Frankle and Carbin, 2019] set the weights below a threshold to zero, rewound the rest of the weights to their initial configuration, and then retrained the network from this configuration.

Though progress has been made, unstructured pruning requires specialized hardware or software supports to speed up inference. It has limited applications on general-purpose hardware or BLAS libraries in practice, due to the irregular sparsity in weight tensors.

Structured Pruning. Compared to unstructured pruning, structured pruning does not have limitations on specialized hardware or software since the entire filters are removed, and thereby it is more favorable in accelerating CNNs.

To this end, regularization-based pruning techniques require a joint-retraining from scratch to derive the values of filters such that they can be made sufficiently small. [Liu *et al.*, 2017; Zhao *et al.*, 2019] imposed a sparse property on the scaling factor of the batch normalization layer with the deterministic l_1-norm or dynamical distribution of channel saliency. After re-training, the channels below the threshold are discarded correspondingly. Differently, [Huang and Wang, 2018] proposed a data-driven sparse structure selection by introducing a scaling factor to scale the outputs of the pruned structure and added the sparsity constraint on the scaling factor. [Lin *et al.*, 2019] proposed to minimize an objective function with l_1-regularization on the soft mask via a generative adversarial learning and adopted the knowledge distillation for optimization in a label-free manner.

Property-based pruning tries to figure out a discriminative property of pre-trained CNN models and discards filters of less importance. [Hu *et al.*, 2016] utilized the abundant zero activations in a large network and iteratively pruned filters with a higher percentage of zero outputs in a layer-wise fashion. [Li *et al.*, 2017] used the sum of absolute values of filters as a metric to measure the importance of filters, and assumed filters with smaller values are less informative and thus should be pruned first. In [Yu *et al.*, 2018], the importance scores of the final responses are propagated to every filter in the network and the CNNs are pruned by removing the one with the least importance.

Optimization-based pruning leverages layer-wise optimization to minimize the reconstruction error between the full model and the pruned model. [He *et al.*, 2017] proposed a LASSO-based filter selection strategy to identify representative filters and a least square reconstruction error to reconstruct the outputs. These two steps are iteratively executed until convergence. [Luo *et al.*, 2017] reconstructed the statistics information from the next layer to guide the importance evaluation of filters from the current layer. A set of training samples is required to deduce a closed-form solution.

The proposed FilterSketch can be grouped into optimization-based pruning but differs from previous methods [He *et al.*, 2017; Luo *et al.*, 2017] in two aspects as shown in Sec. 4: First, it preserves the second-order information of pre-trained weights, leading to quick accuracy recover without the requirement of training from scratch or layer-wise fine-tuning. Second, it can be formulated as the matrix sketch problem and solved by the off-the-shelf Frequent Direction method, leading to a magnitude-order reduction of time consumption without introducing data-driven and/or iterative optimization procedure.

3 The Proposed Approach

In this section, we first introduce a general formulation of the second-order covariance preserving. We then show that network pruning can be converted to the matrix sketch problem and solved by an off-the-shelf sketch method [Liberty, 2013].

3.1 Notations

We start with notation definitions. Given a pre-trained CNN model F, which contains L convolutional layers, and a set of filters $W = \{W^i\}_{i=1}^L$ with $W^i = \{W_j^i\}_{j=1}^{c_i} \in \mathbb{R}^{d_i \times c_i}$ and $d_i = c_{i-1} \times h_i \times w_i$, where c_i, h_i and w_i respectively denote the channel number, filter height and width of the i-th layer. W^i is the filter set for the i-th layer and W_j^i is the j-th filter in the i-th layer.

The goal is to search for the pruned model $\mathcal{F}$, a set of transformed filters $\mathcal{W} = \{\mathcal{W}^i\}_{i=1}^L$ with $\mathcal{W}^i = \{\mathcal{W}_j^i\}_{j=1}^{\tilde{c}_n} \in \mathbb{R}^{d_i \times \tilde{c}_i}$ and $\tilde{c}_i = \lfloor p_i \cdot c_i \rceil$, where p_i is the pruning rate for the i-th layer and $0 < p_i \leq 1$. $\lfloor \cdot \rceil$ rounds the input to its nearest integer. When $p_i < 1$, $c_i - \tilde{c}_i$ filters are pruned. When $p_i = 1$, the number of filters in the i-th layer of the pruned model is kept the same as that of the pre-trained model.

To learn $\mathcal{W}^i$ for each layer, predominant approaches are often divided into three streams: (1) Retraining CNNs from scratch by imposing human-designed regularizations into the training loss [Huang and Wang, 2018; Lin *et al.*, 2019]. (2) Measuring the importance of filters via an intrinsic property of CNNs [Li *et al.*, 2017; Yu *et al.*, 2018]. (3) Minimizing the reconstruction error [He *et al.*, 2017; Luo *et al.*, 2017] for pruning optimization. Nevertheless, these methods solely consider the first-order statistics, which brings less valuable information.

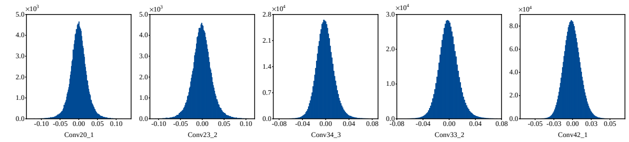


Figure 2: Distributions of weights at different layers of ResNet-50 trained on ILSVRC-2012, which have zero mean.

3.2 Information Preserving

In this study, we devise a novel second-order covariance preserving scheme, which provides a good warm-up for fine-tuning the pruned network. Different from existing work [Kim *et al.*, 2018; Zhang *et al.*, 2019; Ren *et al.*, 2019] where the covariance statistics of feature maps are calculated, we aim to preserve the covariance information of filters, which is inspired by the work [Sun *et al.*, 2017] that reveals the existence of correlation among weights.

For each $W^i \in \mathbb{R}^{d_i \times c_i}$, the second-order preserving scheme aims to find a filter matrix $\mathcal{W}^i \in \mathbb{R}^{d_i \times \tilde{c}_i}$, which contains only $\tilde{c}_i \leq c_i$ columns but preserves sufficient covariance information of W^i, as:

$$\Sigma_{W^i} \approx \Sigma_{\mathcal{W}^i}, \tag{1}$$

where Σ_{W^i} and $\Sigma_{\mathcal{W}^i}$ respectively denote the covariance matrixces of W^i and $\mathcal{W}^i$ and are defined as:

$$\Sigma_{W^i} = (W^i - \bar{W}^i)(W^i - \bar{W}^i)^T, \tag{2}$$

$$\Sigma_{\mathcal{W}^i} = (\mathcal{W}^i - \bar{\mathcal{W}}^i)(\mathcal{W}^i - \bar{\mathcal{W}}^i)^T, \tag{3}$$

where $\bar{W}^i = \frac{1}{c_i}\sum_{j=1}^{c_i} W_j^i$ and $\bar{\mathcal{W}}^i = \frac{1}{\tilde{c}_i}\sum_{j=1}^{\tilde{c}_i} \mathcal{W}_j^i$ are the mean values of the filters in the i-th layer for the full model and pruned model, respectively.

To analyze, it is observed that the covariance Σ_{W^i} can effectively measure the pairwise interactions between filters in the pre-trained model. A key ingredient of our approach is that it can well preserve the correlation information of W^i in $\mathcal{W}^i$. Through this, it yields a more expressive and informative $\mathcal{W}^i$, which then serves as a good warm-up for fine-tuning the pruned model, as validated in Sec. 4.

To achieve the goal of covariance information preserving in Eq. 1, we formulate the following objective function:

$$\arg\min_{\mathcal{W}^i} \|\Sigma_{W^i} - \Sigma_{\mathcal{W}^i}\|_F, \tag{4}$$

where $\|\cdot\|_F$ denotes the Frobenius norm. Based on Eq. 2 and Eq. 3, Eq. 4 is expanded as:

$$\begin{aligned}\arg\min_{\mathcal{W}^i} \big\| &(W^i - \bar{W}^i)(W^i - \bar{W}^i)^T \\ &- (\mathcal{W}^i - \bar{\mathcal{W}}^i)(\mathcal{W}^i - \bar{\mathcal{W}}^i)^T \big\|_F.\end{aligned} \tag{5}$$

We empirically find that, for a pre-trained model, the mean values of filters in each layer approximate to zero, *i.e.*, $\bar{W}^i \approx 0$, as shown in Fig. 2. To explain, the widely-used regularization term (*e.g.*, l_1-norm and l_2-norm) leads to a sparse weight distribution. Similarly, it is intuitive that a good pruned weight $\mathcal{W}^i$ should satisfy that $\bar{\mathcal{W}}^i \approx 0$. Accordingly, Eq. 1 is reformulated as:

$$\arg\min_{\mathcal{W}^i} \big\| W^i (W^i)^T - \mathcal{W}^i (\mathcal{W}^i)^T \big\|_F. \tag{6}$$

Algorithm 1 Frequent Direction [Liberty, 2013].

Require: Data matrix $W^i \in \mathbb{R}^{d_i \times c_i}$, sketched size $\tilde{c}_i$.
Ensure: Sketched matrix $\mathcal{W}^i \in \mathbb{R}^{d_i \times \tilde{c}_i}$.
1: $\mathcal{W}^i \leftarrow$ all zeros matrix $\in \mathbb{R}^{d_i \times \tilde{c}_i}$.
2: **for** each column W_j^i in W^i **do**
3: Insert W_j^i into a zero valued column of $\mathcal{W}^i$.
4: **if** $\mathcal{W}^i$ has no zero valued columns **then**
5: $[U, S, V] = SVD(\mathcal{W}^i)$
6: $\delta = s^2_{\frac{\tilde{c}_i}{2}}$. # The squared $(\frac{\tilde{c}_i}{2})$-th entry of S.
7: $\hat{S} = \sqrt{\max(S^2 - I_{\tilde{c}_i}\delta, 0)}$. # Identity matrix $I_{\tilde{c}_i} \in \mathbb{R}^{\tilde{c}_i \times \tilde{c}_i}$.
8: $\mathcal{W}^i = U\hat{S}$. # At least half the columns of $\mathcal{W}^i$ are all zero.
9: **end if**
10: **end for**

Algorithm 2 FilterSketch Algorithm.

Require: Pre-trained model F with filter set $W = \{W^i\}_{i=1}^L$.
Ensure: Pruned model $\mathcal{F}$ with filter set $\mathcal{W} = \{\mathcal{W}^i\}_{i=1}^L$.
for $i = 1 \rightarrow L$ **do**
Obtain $\mathcal{W}^i$ via sketching W^i by Algorithm 1.
Normalize $\mathcal{W}^i$ via l_2-norm.
end for
Initialize $\mathcal{F}$ with the sketched filter set $\mathcal{W}$.
for $t = 1 \rightarrow T$ **do**
Fine-tune the pruned model $\mathcal{F}$.
end for
Return $\mathcal{F}$ with the fine-tuned filter set $\mathcal{W}$.

The goal of covariance information preserving in Eq. 1 is transformed to optimize the objective function defined in Eq. 6. To optimize Eq. 6, similar to [He *et al.*, 2017; Luo *et al.*, 2017], one can develop a series of optimization steps to minimize the reconstruction error of Eq. 6. However, the optimization procedure is typically based on data-driven and/or iterative methods [He *et al.*, 2017; Luo *et al.*, 2017], which inevitably introduce heavy computation cost.

3.3 Tractability

We show that Eq. 6 can be effectively and efficiently solved by the off-the-shelf matrix sketch method [Liberty, 2013], which does not involve data-driven iterative optimization while maintaining the property of interest of W^i in $\mathcal{W}^i$.

Specifically, a sketch of a matrix W^i is a transformed matrix $\mathcal{W}^i$, which is smaller than W^i but tracks an ε-approximation to the norm of W^i, as:

$$\begin{aligned} &\mathcal{W}^i(\mathcal{W}^i)^T \preccurlyeq W^i(W^i)^T, \text{ and} \\ &\left\| W^i(W^i)^T - \mathcal{W}^i(\mathcal{W}^i)^T \right\|_F \leq \varepsilon \|W^i\|_F^2. \end{aligned} \tag{7}$$

Several arts have been devoted to the matrix sketch problem, including CUR decomposition [Drineas and Kannan, 2003], random projection [Sarlos, 2006], and column sampling methods [Frieze *et al.*, 2004], which however still rely on iterative optimization.

The streaming-based Frequent Direction (FD) method by [Liberty, 2013] provides a promising direction to solve this problem, where each sample is passed forward only once and we summarize it in Alg. 1. A $d_i \times c_i$ data matrix W^i and the sketched size $\tilde{c}_i$ are fed into FD. Each column W_j^i of matrix W^i represents a sample. Columns from W^i will replace all zero-valued columns in $\mathcal{W}^i$, and half of the columns in the sketch will be emptied with two steps once $\mathcal{W}^i$ is fully fed with non-zero valued columns: In the first step, the sketch is rotated (from right) with the SVD decomposition of $\mathcal{W}^i$ so that its columns are orthogonal and in descending magnitude order. In the SVD decomposition, $USV^T = \mathcal{W}^i$, $U^TU = V^TV = VV^T = I_{\tilde{c}_i}$, where $I_{\tilde{c}_i}$ is the $\tilde{c}_i \times \tilde{c}_i$ identity matrix, S is a non-negative diagonal matrix and $S_{11} \geq ... \geq S_{\tilde{c}_i\tilde{c}_i} \geq 0$. In the second step, S is shrunk so that half of its singular values are zeros. Accordingly, the right half of the columns in $U\hat{S}$ (see line 7 of Alg. 1 for $\hat{S}$) will be zeros. The details of the method can be referred to [Liberty, 2013].

It can be seen that the optimization of Eq. 6 is similar to the matrix sketch problem of Eq. 7, though the existence of the upper bound term $\varepsilon\|W^i\|_2^2$ does not necessarily result in optimal $\mathcal{W}^i$ for Eq. 6. Nevertheless, in what follows, we show that Alg. 1 can provide a tight convergence bound to solve the sketch problem of Eq. 7 while the learned $\mathcal{W}^i$ can serve as a good warm-up for fine-tuning the pruned model as demonstrated in Sec. 4.

Corollary 1. If $\mathcal{W}^i$ is the sketch result of matrix W^i with the sketched size $\tilde{c}_i$ by Alg. 1, then the following holds:

$$\begin{aligned} &0 \preccurlyeq \mathcal{W}^i(\mathcal{W}^i)^T \preccurlyeq W^i(W^i)^T, \text{ and} \\ &\left\| W^i(W^i)^T - \mathcal{W}^i(\mathcal{W}^i)^T \right\|_F \leq \frac{2}{\tilde{c}_i} \|W^i\|_F^2, \end{aligned} \tag{8}$$

i.e., $\varepsilon = \frac{2}{\tilde{c}_i}$ and the proof can be referred to [Liberty, 2013].

Accordingly, the convergence bound of FD is proportional to $\frac{1}{\tilde{c}_i}$. Smaller $\tilde{c}_i$ causes more error, which is intuitive since smaller $\tilde{c}_i$ means more pruned filters. Besides, the sketch time is up-bounded by $\mathcal{O}(d_i c_i \tilde{c}_i)$ [Liberty, 2013]. Sec. 4.5 shows that the sketch process requires less than 2 seconds in CPU implementation, which verifies the efficiency of FilterSketch.

In our implementation of FilterSketch, l_2-norm is applied to $\mathcal{W}^i$ before it is fed to the slimmed network for fine-tuning. In what follows, we provide a tighter convergence bound and Sec. 4.5 verifies its ability to boost the performance.

Theorem 1. If $\mathcal{W}^i$ is the sketch result by applying Alg. 1 to matrix W^i with the sketched size $\tilde{c}_i$, then for any constant β, $\beta\mathcal{W}^i$ is the result by applying Alg. 1 to matrix βW^i.

Proof. To prove **Theorem 1**, we start with Line 5 of Alg. 1, which can be modified as:

Line 5: $[U, \beta S, V] = SVD(\beta\mathcal{W}^i)$.

Correspondingly, Line 6 to Line 8 can be modified as:

Line 6: $\beta^2\delta = (\beta s_{\frac{\tilde{c}_i}{2}})^2$;

Line 7: $\beta\hat{S} = \sqrt{\max\left(\beta^2 S^2 - I_{\tilde{c}_i}\beta^2\delta, 0\right)}$;

Line 8: $\beta\mathcal{W}^i = U\beta\hat{S}$.

Thus, the sketch of βW^i results in $\beta\mathcal{W}^i$, which completes the proof. ■

According to **Corollary 1**, we have:

$$\begin{aligned} &\left\| (\beta W^i)(\beta W^i)^T - (\beta\mathcal{W}^i)(\beta\mathcal{W}^i)^T \right\|_F \\ &\leq \frac{2}{\tilde{c}_i}\|\beta W^i\|_F^2 = \frac{2\beta^2}{\tilde{c}_i}\|W^i\|_F^2. \end{aligned} \tag{9}$$

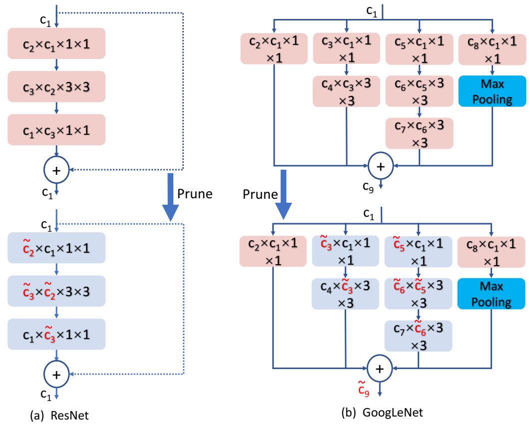


Figure 3: FilterSketch on ResNet and GoogLeNet. For ResNet, we prune the first two convolutional layers in each block. For GoogLeNet, we prune the branches with more than one convolutional layer and meanwhile the last convolutional layer is kept unchanged. For all pruned filters, the channels in the next layer are removed correspondingly. (Best viewed with zooming in)

Practically, we set $\beta = \frac{1}{\|\mathcal{W}^i\|_2}$. Note that $\beta < 1$ is always satisfied from our extensive experiments. Then

$$\frac{2\beta^2}{\tilde{c}_i}\|W^i\|_F^2 < \frac{2}{\tilde{c}_i}\|W^i\|_F^2, \tag{10}$$

which shows that the sketch of βW^i ($\beta = \frac{1}{\|\mathcal{W}^i\|_2}$) offers a tighter bound of the reconstruction error than that of W^i. Hence, using $\frac{\mathcal{W}^i}{\|\mathcal{W}^i\|}$ is more approximate to $\frac{W^i}{\|\mathcal{W}^i\|}$ than W^i to $\mathcal{W}^i$ and its effectiveness is shown in Sec. 4.5[1].

The proposed FilterSketch is outlined in Alg. 2. It can be seen that, compared with existing methods, FilterSketch stands out in that it is deterministic and simple to implement.

4 Experiments

To show the effectiveness and efficiency of FilterSketch, we have conducted extensive experiments on image classification. Representative compact-designed networks, including GoogLeNet [Szegedy *et al.*, 2015] and ResNet-50/56/110 [He *et al.*, 2016], are chosen to compress. We report the performance of FilterSketch on CIFAR-10 [Krizhevsky *et al.*, 2009] and ILSVRC-2012 [Russakovsky *et al.*, 2015], and compare it to state-of-the-arts including regularization-based pruning [Huang and Wang, 2018; Lin *et al.*, 2019], property-based pruning [Li *et al.*, 2017; Yu *et al.*, 2018], and optimization-based pruning [He *et al.*, 2017; Luo *et al.*, 2017]. Fig. 3 outlines our pruning strategy for ResNet and GoogLeNet.

4.1 Implementation Details

Training Strategy. We use the Stochastic Gradient Descent algorithm (SGD) for fine-tuning with Nesterov momentum 0.9 and the batch size is set to 256. For CIFAR-10, the weight decay is set to 5e-3 and we fine-tune the network for 150 epochs with an initial learning rate of 0.01, which is then divided by 10 every 50 epochs. For ILSVRC-2012, the weight decay is set to 5e-4 and 90 epochs are given to fine-tune the network. The learning rate is initially set to 0.1, and divided by 10 every 30 epochs.

Performance Metric. Parameter amount and FLOPs (floating-point operations) are used as metrics, which respectively denote the storage and computation cost. We also report the pruning rate (PR) of parameters and FLOPs. For CIFAR-10, top-1 accuracy are provided. For ILSVRC-2012, both top-1 and top-5 accuracies are reported.

Model	Top-1%	FLOPs(PR)	Parameters(PR)
GoogLeNet	95.05	1.52B(0.0%)	6.15M(0.0%)
Random	94.54	0.96B(36.8%)	3.58M(41.8%)
L1 [Li *et al.*, 2017]	94.54	1.02B(32.9%)	3.51M(42.9%)
GAL-0.05 [Lin *et al.*, 2019]	93.93	0.94B(38.2%)	3.12M(49.3%)
FilterSketch (Ours)	94.88	0.59B(61.1%)	2.61M(57.6%)

Table 1: Pruning results of GoogLeNet on CIFAR-10.

Model	Top-1%	FLOPs(PR)	Parameters(PR)
ResNet-56	93.26	125.49M(0.0%)	0.85M(0.0%)
L1 [Li *et al.*, 2017]	93.06	90.90M(27.6%)	0.73M(14.1%)
NISP [Yu *et al.*, 2018]	93.01	81.00M(35.5%)	0.49M(42.4%)
GAL-0.6 [Lin *et al.*, 2019]	92.98	78.30M(37.6%)	0.75M(11.8%)
FilterSketch (Ours)	93.19	73.36M(41.5%)	0.50M(41.2%)

Table 2: Pruning results of ResNet-56 on CIFAR-10.

Model	Top-1%	FLOPs(PR)	Parameters(PR)
ResNet-110	93.50	252.89M(0.0%)	1.72M(0.0%)
L1 [Li *et al.*, 2017]	93.30	155.00M(38.7%)	1.16M(32.6%)
GAL-0.5 [Lin *et al.*, 2019]	92.55	130.20M(48.5%)	0.95M(44.8%)
FilterSketch (Ours)	93.44	92.84M(63.3%)	0.69M(59.9%)

Table 3: Pruning results of ResNet-110 on CIFAR-10.

4.2 Results on CIFAR-10

We evaluate the performance of FilterSketch on CIFAR-10 with popular networks, including GoogLeNet, ResNet-56 and ResNet-110. For GoogLeNet, we change the final output class number as the number of categories on CIFAR-10.

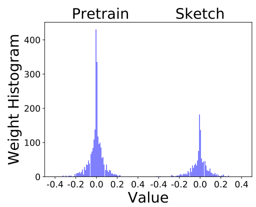
GoogLeNet. Tab. 1 shows that FilterSketch outperforms the state-of-the-art methods in both accuracy retaining and model complexity reductions. Specifically, 61.1% of the FLOPs are reduced and 57.6% of the parameters are removed, achieving a significantly higher compression rate than GAL-0.6. Besides, FilterSketch also maintains a comparable top-1 accuracy, even better than L1, which obtains a much less complexity reduction.

ResNet-56. Results for ResNet-56 are presented in Tab. 2, where FilterSketch removes around 41.5% of the FLOPs and parameters while keeping the top-1 accuracy at 93.19%. Compared to 93.26% by the pre-trained model, the accuracy drop is negligible. As for L1, FilterSketch shows an overwhelming superiority. Though NISP obtains 1% more parameters reduction than FilterSketch, it takes more computation

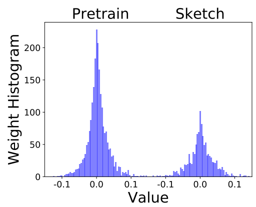
[1]Another direction is to sketch $\frac{W^i}{\|W^i\|_2}$, which will return $\frac{\mathcal{W}^i}{\|W^i\|_2}$ as the sketch result. However, $\|W^i\|_2$ is much larger than $\|\mathcal{W}^i\|_2$, leading $\frac{\mathcal{W}^i}{\|W^i\|_2}$ to an almost zero matrix, which is infeasible.

Model	Top-1%	Top-5%	FLOPs(PR)	Parameters(PR)
ResNet-50 [Luo *et al.*, 2017]	76.13	92.86	4.09B(0.0%)	25.50M(0.0%)
SSS-32 [Huang and Wang, 2018]	74.18	91.91	2.82B(31.1%)	18.60M(27.1%)
He *et al.* [He *et al.*, 2017]	72.30	90.80	2.73B(33.3%)	-
FilterSketch-0.7 (Ours)	75.22	92.41	2.64B(35.5%)	16.95M(33.5%)
GAL-0.5 [Lin *et al.*, 2019]	71.95	90.94	2.33B(43.0%)	21.20M(16.9%)
SSS-26 [Huang and Wang, 2018]	71.82	90.79	2.33B(43.0%)	15.60M(38.8%)
FilterSketch-0.6 (Ours)	74.68	92.17	2.23B(45.5%)	14.53M(43.0%)
GAL-0.5-joint [Lin *et al.*, 2019]	71.80	90.82	1.84B(55.0%)	19.31M(24.3%)
ThiNet-50 [Luo *et al.*, 2017]	71.01	90.02	1.71B(58.2%)	12.28M(51.8%)
GAL-1 [Lin *et al.*, 2019]	69.88	89.75	1.58B(61.4%)	14.67M(42.5%)
FilterSketch-0.4 (Ours)	73.04	91.18	1.51B(63.1%)	10.40M(59.2%)
ThiNet-50 [Luo *et al.*, 2017]	68.42	88.30	1.10B(73.1%)	8.66M(66.0%)
GAL-1-joint [Lin *et al.*, 2019]	69.31	89.12	1.11B(72.9%)	10.21M(60.0%)
FilterSketch-0.2 (Ours)	69.43	89.23	0.93B(77.3%)	7.18M(71.8%)

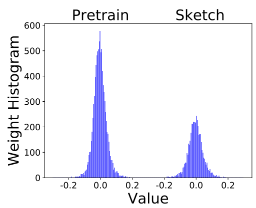
Table 4: Pruning results of ResNet-50 on ILSVRC-2012.



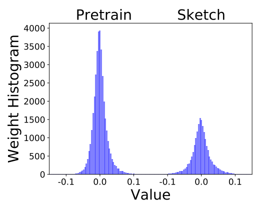
(a) Layer1_Block0_Conv1.



(b) Layer1_Block7_Conv2.



(c) Layer2_Block5_Conv1.



(d) Layer3_Block8_Conv2.

Figure 4: Distribution comparisons between weights with and without filter sketch for ResNet-56.

in the convolutional layers with a lower top-1 accuracy.

ResNet-110. Tab. 3 also displays the pruning results for ResNet-110. FilterSketch reduces the FLOPs of ResNet-110 by an impressive factor of 63.3%, and the parameters by 59.9%, while maintaining an accuracy of 93.44%. FilterSketch significantly outperforms state-of-the-art methods, which shows that it greatly facilitates the ResNet model, a popular backbone for object detection and semantic segmentation, to be deployed on mobile devices.

4.3 Results on ILSVRC-2012

In Tab. 4, we show the results for ResNet-50 on ILSVRC-2012 and compare FilterSketch to many SOTAs. We display different pruning rates for FilterSketch and compare top-1 and top-5 accuracies. For convenience, we use FilterSketch-α to denote the sketch rate, *i.e.*, $\alpha = \frac{\tilde{c}}{c}$. As can be seen, smaller α leads to a higher compression ratio.

As can be seen from Tab. 4, with similar or better reductions of FLOPs and parameters, FilterSketch displays its great advantages in retaining the accuracy performance in comparisons with SOTAs. For example, FilterSketch-0.6 obtains 74.68% top-1 and 92.17% top-5 accuracies, significantly better than GAL-0.5 and SSS-26. Another observation is that, with similar or more FLOPs reduction, FilterSketch also removes a greater number of parameters. Hence, FilterSketch is especially suitable for network compression.

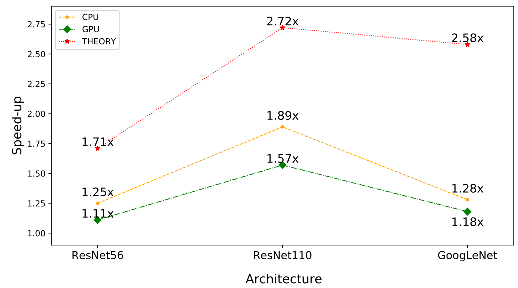


Figure 5: Speedups corresponding to CPU (Intel(R) Xeon(R) CPU E5-2620 v4 @2.10GHz) and GPU (GTX-1080TI) over the different CNNs with a batch size 256 on CIFAR-10.

Analysis. From Sec. 4.2 and Sec. 4.3, we can see that FilterSketch presents a better ability to relieve the CNN complexity. For an in-depth analysis, we compare the weights distributions between a pre-trained model and its sketched model, and show some examples in Fig. 4. As can be seen, the weights with and without sketch fall into nearly identical value intervals in each sub-figure. Hence, FilterSketch can well retain the second-order information of the pre-trained model, which provides a more expressive and informative warm-up for fine-tuning the pruned model.

4.4 Practical Speedup

The practical speedup for pruned CNNs depends on many factors, *e.g.*, FLOPs reduction percentage, the number of CPU/GPU cores available and I/O delay of data swap, *etc*.

We test the speedup of pruned models by FilterSketch in Tab. 1, Tab. 2, and Tab. 3 with CPU and GPU, and present the results in Fig. 5. Compared with the theoretical speedups of 1.71×, 2.72× and 2.58× for ResNet-56, ResNet-110 and GoogLeNet, respectively. FilterSketch gains 1.11×, 1.57× and 1.18× practical speedups with GPU while 1.25×, 1.89× and 1.28× with CPU are obtained.

4.5 l_2-norm Influence and Optimization Efficiency

To measure the effectiveness of the sketch with l_2-norm, we compare our FilterSketch models given in Tab. 1, Tab. 2, Tab. 3 and Tab. 4 (FilterSketch-0.4) with these models but without l_2-norm. As shown in Tab. 5, the former (the third column) obtains a better accuracy than the latter (the second column), which verifies the analysis in Sec. 3.3 that the sketch with l_2-norm offers a tighter reconstruction error of the second-order covariance. As for the sketch efficiency, we again compare these four models given in Tabs. 1–4 with two optimization-based methods [Luo *et al.*, 2017; He *et al.*, 2017]. The results in Tab. 6 show that the time cost on sketch process is minor. Even with wider GoogLeNet and deeper ResNet-110, the sketch consumes less than 2 seconds that are negligible compared with the other methods, which cost many hours, or even days.

	Sketch accuracy	Sketch+l_2 accuracy
GoogLeNet	94.54%	94.88%
ResNet-56	92.81%	93.19%
ResNet-110	93.01%	93.44%
ResNet-50	72.84% / 91.01%	73.04% / 91.18%

Table 5: Performance comparisons between sketch with and without l_2-norm.

	ThiNet	CP	**FilterSketch (Ours)**
GoogLeNet	183585.41s	2008.72s	1.85s
ResNet-56	24422.73s	536.55s	0.09s
ResNet-110	63695.89s	961.71s	1.06s
ResNet-50	4130102.59s	205117.20s	1.41s

Table 6: Comparisons of optimization efficiency (CPU) among ThiNet [Luo *et al.*, 2017], CP [He *et al.*, 2017] and FilterSketch.

5 Conclusion

We have proposed a novel approach termed FilterSketch for structured network pruning. Instead of simply discarding unimportant filters, FilterSketch preserves the second-order information of the pre-trained model, through which the accuracy drops are quickly recovered. We have further proposed to obtain the information preserving constraint by utilizing the off-the-shelf matrix sketch method, based on which the requirement of training from scratch or iterative optimization can be eliminated, and the pruning complexity is significantly reduced. Extensive experiments on popular CNN architectures demonstrate the superiorities of FilterSketch over the state-of-the-arts. As the first attempt on weight information preserving, FilterSketch provides a fresh insight for the network pruning problem.

6 Acknowledge

This work is supported by the Nature Science Foundation of China (No.U1705262, No.61772443, No.61572410, No.61802324 and No.61702136), National Key R&D Program (No.2017YFC0113000, and No.2016YFB1001503), and Nature Science Foundation of Fujian Province, China (No. 2017J01125 and No. 2018J01106).

References

[Alvarez and Salzmann, 2016] Jose M Alvarez and Mathieu Salzmann. Learning the number of neurons in deep networks. In *NeurIPs*, 2016.

[Denil *et al.*, 2013] Misha Denil, Babak Shakibi, Laurent Dinh, Marc'Aurelio Ranzato, and Nando De Freitas. Predicting parameters in deep learning. In *NeurIPs*, 2013.

[Dong *et al.*, 2017] Xin Dong, Shangyu Chen, and Sinno Pan. Learning to prune deep neural networks via layer-wise optimal brain surgeon. In *NeurIPS*, 2017.

[Drineas and Kannan, 2003] Petros Drineas and Ravi Kannan. Pass efficient algorithms for approximating large matrices. In *SODA*, 2003.

[Frankle and Carbin, 2019] Jonathan Frankle and Michael Carbin. The lottery ticket hypothesis: Finding sparse, trainable neural networks. In *ICLR*, 2019.

[Frieze *et al.*, 2004] Alan Frieze, Ravi Kannan, and Santosh Vempala. Fast monte-carlo algorithms for finding low-rank approximations. *ACM JACM*, 2004.

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

[Hayashi *et al.*, 2019] Kohei Hayashi, Taiki Yamaguchi, Yohei Sugawara, and Shin-ichi Maeda. Exploring unexplored tensor network decompositions for convolutional neural networks. In *NeurIPS*, 2019.

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

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

[Hu *et al.*, 2016] Hengyuan Hu, Rui Peng, Yu-Wing Tai, and Chi-Keung Tang. Network trimming: A data-driven neuron pruning approach towards efficient deep architectures. *arXiv preprint arXiv:1607.03250*, 2016.

[Huang and Wang, 2018] Zehao Huang and Naiyan Wang. Data-driven sparse structure selection for deep neural networks. In *ECCV*, 2018.

[Hubara *et al.*, 2016] Itay Hubara, Matthieu Courbariaux, Daniel Soudry, Ran El-Yaniv, and Yoshua Bengio. Binarized neural networks. In *NeurIPs*, 2016.

[Kim *et al.*, 2018] Jin-Hwa Kim, Jaehyun Jun, and Byoung-Tak Zhang. Bilinear attention networks. In *NeurIPS*, 2018.

[Krizhevsky *et al.*, 2009] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. Technical report, 2009.

[LeCun *et al.*, 1990] Yann LeCun, John S Denker, and Sara A Solla. Optimal brain damage. In *NeurIPs*, 1990.

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

[Liberty, 2013] Edo Liberty. Simple and deterministic matrix sketching. In *ACM SIGKDD*, 2013.

[Lin *et al.*, 2019] Shaohui Lin, Rongrong Ji, Chenqian Yan, Baochang Zhang, Liujuan Cao, Qixiang Ye, Feiyue Huang, and David Doermann. Towards optimal structured cnn pruning via generative adversarial learning. In *CVPR*, 2019.

[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.

[Liu *et al.*, 2018] Zhenhua Liu, Jizheng Xu, Xiulian Peng, and Ruiqin Xiong. Frequency-domain dynamic pruning for convolutional neural networks. In *NeurIPS*, 2018.

[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.

[Ren *et al.*, 2019] Chuan-Xian Ren, Jiashi Feng, Dao-Qing Dai, and Shuicheng Yan. Heterogeneous domain adaptation via covariance structured feature translators. *IEEE T-Cybernetics*, 2019.

[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.

[Sarlos, 2006] Tamas Sarlos. Improved approximation algorithms for large matrices via random projections. In *FOCS*, 2006.

[Singh *et al.*, 2019] Pravendra Singh, Vinay Kumar Verma, Piyush Rai, and Vinay P. Namboodiri. Play and prune: Adaptive filter pruning for deep model compression. In *IJCAI*, 2019.

[Sun *et al.*, 2017] Yifan Sun, Liang Zheng, Weijian Deng, and Shengjin Wang. Svdnet for pedestrian retrieval. In *CVPR*, 2017.

[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.

[Wang *et al.*, 2019] Kuan Wang, Zhijian Liu, Yujun Lin, Ji Lin, and Song Han. Haq: Hardware-aware automated quantization with mixed precision. In *CVPR*, 2019.

[Yu *et al.*, 2018] Ruichi Yu, Ang Li, Chun-Fu Chen, Jui-Hsin Lai, Vlad I Morariu, Xintong Han, Mingfei Gao, Ching-Yung Lin, and Larry S Davis. Nisp: Pruning networks using neuron importance score propagation. In *CVPR*, 2018.

[Zhang *et al.*, 2019] Jianhai Zhang, Zhiyong Feng, Yong Su, and Meng Xing. Discriminative saliency-pose-attention covariance for action recognition. In *ICASSP*, 2019.

[Zhao *et al.*, 2019] Chenglong Zhao, Bingbing Ni, Jian Zhang, Qiwei Zhao, Wenjun Zhang, and Qi Tian. Variational convolutional neural network pruning. In *CVPR*, 2019.