Linearly Replaceable Filters for Deep Network Channel Pruning

Donggyu Joo, Eojindl Yi, Sunghyun Baek, Junmo Kim
School of Electrical Engineering, KAIST
{jdg105, djwld93, baeksh, junmo.kim}@kaist.ac.kr

Abstract

Convolutional neural networks (CNNs) have achieved remarkable results; however, despite the development of deep learning, practical user applications are fairly limited because heavy networks can be used solely with the latest hardware and software supports. Therefore, network pruning is gaining attention for general applications in various fields. This paper proposes a novel channel pruning method, Linearly Replaceable Filter (LRF), which suggests that a filter that can be approximated by the linear combination of other filters is replaceable. Moreover, an additional method called Weights Compensation is proposed to support the LRF method. This is a technique that effectively reduces the output difference caused by removing filters via direct weight modification. Through various experiments, we have confirmed that our method achieves state-of-the-art performance in several benchmarks. In particular, on ImageNet, LRF-60 reduces approximately 56% of FLOPs on ResNet-50 without top-5 accuracy drop. Further, through extensive analyses, we proved the effectiveness of our approaches.

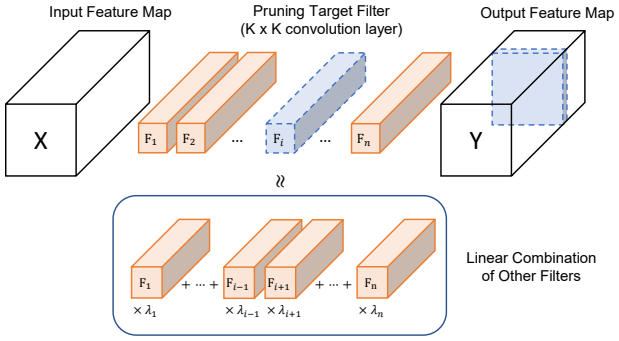

Figure 1: Basic idea of proposed linearly replaceable filters (LRF). At every layer, each filter can be approximated by the linear combination of other $n-1$ filters. If F_i is the most replaceable one which has the lowest approximation error, the network can be recovered even if it is removed.

Introduction

Computer vision has been revolutionized with the advent of convolutional neural networks (CNNs) such as AlexNet (Krizhevsky, Sutskever, and Hinton 2012), VGG (Simonyan and Zisserman 2014), and ResNet (He et al. 2016a). Subsequently, even heavier networks (Huang et al. 2017; Xie et al. 2017) were designed to achieve higher performance. Advances in technology have enabled the enormous computation these networks require. However, in some environments like mobile devices, the computational resources are limited, and high speed is often required. Therefore, the practical scope and everyday usage of deep learning are extremely limited. For the widespread use of artificial intelligence, we need method that makes networks more compact while preserving most of their accuracies.

Recently, the network pruning field has been gaining attention. Given a pre-trained network, the goal of network pruning is to make this network more compact by reducing redundant parameters and operations. Network pruning began with removing unnecessary weights (Han et al. 2015) from the deep neural networks (DNNs). However, because each channel of the CNNs is connected with a large number of weights, removing individual weights does not necessarily lead to the actual speedup of the CNNs. Thus, channel pruning that removes a specific channel and all its connected weights together is being researched more actively (He et al. 2019; You et al. 2019; Zhuang et al. 2018).

In this paper, we propose a novel channel pruning method called *Linearly Replaceable Filter (LRF)*, that considers the linear replaceability of each filter. For a specific layer, there are n different filters. As shown in Figure 1, we can approximate each filter as a linear combination of the other $n-1$ filters. If this approximation for a certain filter is accurate enough with a low approximation error, this filter is considered to be replaceable, and the network can be easily restored when this filter is pruned. These replaceable filters have the advantage that other remaining filters can easily replace their role in the later fine-tuning process.

The proposed method does not end with simply identifying and removing the filters that are expected to be replaceable. In addition to LRF, we propose an output difference compensation technique called *Weights Compensation* to support LRF. Using the property of 1×1 convolution, we can effectively reduce the difference in output feature maps caused by removing filters. When removing each channel,

we directly modify the values of the remaining weights to the appropriate values such that the network can be preserved. This process does not require any extra fine-tuning or learning, and the difference is drastically reduced through simple operations. Most existing studies have attempted to identify the least important filter and simply remove it or design a special loss to make a particular filter less important. However, the proposed method directly changes the value of the remaining weight for the successful channel pruning. This approach provides a clear theoretical basis, unlike traditional pruning methods. We have achieved state-of-the-art results when the proposed LRF and Weights Compensation are used together.

Moreover, we conducted various analyses of LRF and the network pruning field. We demonstrate the effectiveness of the proposed method and also discuss the problem of accuracy drop, an important metric in the field of pruning. The key contributions of our work are summarized as follows:

1) We propose a novel perspective of channel pruning called *Linearly Replaceable Filter (LRF)* that the network can be successfully pruned with the replaceable filter that can be expressed by a linear combination of other filters.

2) By directly modifying the values of remaining filters after eliminating each channel, we can reduce the difference occurring in the output feature maps. We call this *Weights Compensation*.

3) We have conducted several analyses of network pruning. In particular, we revealed that considering the accuracy drop alone could be problematic when evaluating the performance of pruning.

Related Work

Network Pruning. Network pruning aims to remove the unnecessary weights of a pre-trained neural network while maintaining high performance. Early pruning (Han et al. 2015) began with the removal of weights or nodes with small-norm from the DNN; however, as CNN evolved, the pruning of a single weight or node was not practically effective. Thus, channel pruning or filter pruning became the mainstream in this field. Several heuristic methods that use the average percentage of zero activations (Hu et al. 2016) and Lasso regression (He, Zhang, and Sun 2017; Luo, Wu, and Lin 2017) were proposed at the early stage of channel pruning research. Since then, various approaches such as regularizing the scaling factor of BatchNorm (BN) (Ioffe and Szegedy 2015) to prune corresponding channel (Liu et al. 2017), measuring the importance score through backprop (Yu et al. 2018), or finding important channels by adding a new loss to the intermediate layer (Zhuang et al. 2018) have been proposed. Recently, more unique methods have emerged (He et al. 2019; You et al. 2019; Lin et al. 2020), such as investigating the value of pruning (Liu et al. 2018) or using meta-learning (Liu et al. 2019).

Other Methods. Knowledge distillation (Hinton, Vinyals, and Dean 2015; Romero et al. 2014; Yim et al. 2017) is a technique of distilling information from a large network and delivering it to a small network. Using this, a small network can be trained significantly better than training from scratch. Apart from using a pre-trained network, several studies (Howard et al. 2017; Tan and Le 2019; Zhang et al. 2018) have been conducted to design an efficient architecture away from the conventional CNN structure either in theoretical or in automatic way. It should be noted that these *other methods* are complementary to pruning and our LRF although their goals look similar. The LRF can be further combined with methods in this subsection to achieve more compact network. Therefore, these methods and LRF are in a good win-win relationship, not in a comparative relationship.

Method

Preliminaries. We aim to reduce the computational complexity of a pre-trained network by removing redundant weights or filters. First, a pre-trained network $\phi(\cdot)$ (we also call it as an original network) is provided. For a specific convolutional layer with kernel size $K \times K$, the weight $\mathcal{F}$ has a dimension of $K \times K \times m \times n$ where m and n represent the number of input and output channels, respectively. The lower-cased $f_{i,j} \in \mathbb{R}^{K \times K}$ denotes the weights that connect the i-th input channel to the j-th output channel. We denote the collection of weights connected to the j-th output channel as $f_{:,j}$, and collection of weights connected to the i-th input channel as $f_{i,:}$ for notational convenience.

Our channel pruning proceeds by iteratively removing one channel of a layer at a time, and stops when the desired number of channels are pruned. When pruning of one layer is finished, the pruning of the next layer proceeds sequentially. In the following section, we explain how we find the most replaceable filter using the concept of LRF. In the next section, we discuss the *weights compensation*, one of our key contributions. By applying weights compensation, we finally present a complete version of our channel selection criterion.

Linearly Replaceable Filters

Given any $K \times K$ convolution layer ($K \in \mathbb{N}$) of pre-trained network $\phi(\cdot)$, this layer has a weight $\mathcal{F} \in \mathbb{R}^{K \times K \times m \times n}$, where m and n represent the number of input and output channels, respectively. The $\mathcal{F}$ can be considered as a collection of n filters $f_{:,j} \in \mathbb{R}^{K \times K \times m}$ for $j = 1, \cdots, n$. Then, each filter $f_{:,j}$ can be approximated as a linear combination of the other $n-1$ filters of the same layer. For all $j = 1, \cdots, n$, this approximation can be expressed using the following equation:

$$f_{:,j} = \sum_{l \neq j} \lambda_{j,l} f_{:,l} + \epsilon_j \tag{1}$$

where ϵ_j is an approximation error with the same dimensions as filter $f_{:,j}$, and $\lambda_{j,l}$ is a scalar coefficient. Because our goal is to best approximate each $f_{:,j}$, we select $\lambda_{j,l}$ that minimizes $||\epsilon_j||$. The coefficients $\lambda_{j,l}$ can be obtained by solving the following problem:

$$\min L_j = \min ||f_{:,j} - \sum_{l \neq j} \lambda_{j,l} f_{:,l}||^2. \tag{2}$$

First we reshape each filter $f_{:,j}$ as a vector with length $K \cdot K \cdot m$. Then, by differentiating with respect to each $\lambda_{j,l'}$ for all l', we obtain the following equation:

$$\frac{\partial L_j}{\partial \lambda_{j,l'}} = 2f_{:,l'}^T(f_{:,j} - \sum_{l \neq j} \lambda_{j,l} f_{:,l}) = 0. \quad (3)$$

Then, it becomes a system of linear equations with $n-1$ variables and $n-1$ equations, and it can be solved by matrix computation. After we obtain all the λ, we can easily calculate all the approximation errors $\epsilon_1, \cdots, \epsilon_n$. Our channel selection criterion for pruning is to pick the channel that corresponds to the most replaceable filter. Therefore, we choose a filter $f_{:,s}$ that has the smallest approximation error ϵ_s, *i.e.*, we choose to remove the filter $f_{:,s}$ such that

$$s = \arg\min_j ||\epsilon_j|| \quad (4)$$

Because we are aiming for channel pruning, removing filter $f_{:,s}$ naturally leads to removing the s-th output channel of this layer and all the associated weights simultaneously. When pruning is completed, every operation is conducted only using the remaining channels. We call this method as Linearly Replaceable Filter (LRF); it means that we select the filters that can be replaced by the linear combination of other filters. Because the pruning process usually does not simply finish with only removing unnecessary channels, but fine-tuning proceeds with the remaining filters, it is very important to identify how the deleted filter is related to the other filters. Removing the replaceable filter can preserve the performance of the original network because the remaining filters can replace the role of the removed filter. Since this step requires only some of simple matrix computations, the time consumption for this process is negligible.

Weights Compensation

Although LRF provides the intuition that replaceability can be beneficial, it does not provide an explicit explanation of whether this can actually lead to successful pruning. Therefore, we propose a supplement method *weights compensation* that makes our method more reliable.

We start by adding a 1×1 convolution on top of the $K\times K$ convolution layer that we want to prune. This 1×1 convolution is initialized as an identity matrix such that the computation of the original network is not affected. This addition of 1×1 convolution enables two things: First, when removing the channels, the change in the network resulting from pruning can be significantly reduced by modifying the remaining weights of this 1×1 convolution. Second, it enables the pruning of any $K \times K$ convolution regardless of the network structure, such as ResNet's skip-connection. One thing to note is that when we add this new 1×1 convolution, any function such as BN (Ioffe and Szegedy 2015) or ReLU is not placed between the two convolutions.

Let $X = \{X_1, \cdots, X_m\}$ and $Y = \{Y_1, \cdots, Y_n\}$ be the input and output of given $K \times K$ convolution. Then, Y becomes the input of the 1×1 convolution layer we added on top of this $K \times K$ convolution. Further, let $Z = \{Z_1, \cdots, Z_n\}$ be the output of this 1×1 convolution layer. Each X_i indicates the i-th channel of feature map X. The number of channels of Y starts with n at the beginning, and it decreases as pruning progresses. The computation of these two convolution layers are as follows:

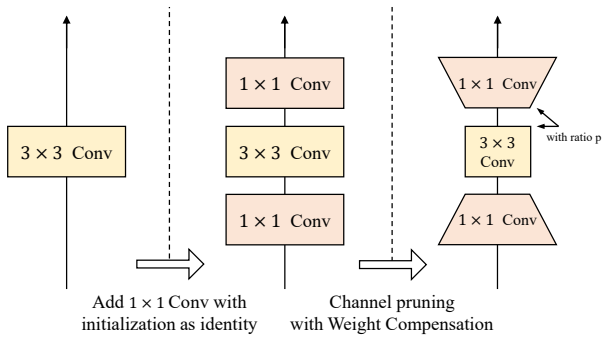

Figure 2: Basic framework of LRF. In the illustration, the width of the box implies the number of channels. A narrow width implies that the channel is pruned with a ratio p. Given any pre-trained network, first, we add two 1×1 convolution layers at the bottom and top of every $K \times K$ convolution layers in the network. Every 1×1 convolution layers are initialized as an identity matrix that does not affect the computation of the pre-trained network. Subsequently, redundant filters are pruned following the criterion of LRF with weights compensation.

$$Y_j = \sum_{i=1}^{m} X_i * f_{i,j} \coloneqq X * f_{:,j} \quad (5)$$

$$Z = \sum_{j=1}^{n} Y_j * g_{j,:} = \sum_{j=1}^{n} X * f_{:,j} * g_{j,:} \quad (6)$$

where $f_{i,j} \in \mathbb{R}^{K\times K}$ is the weight of $K \times K$ convolution that connects X_i and Y_j. Similarly, $g_{j,k} \in \mathbb{R}$ is the weight of 1×1 convolution that connects Y_j and Z_k. If we remove, for example, the first channel Y_1 of Y, then the output feature map Z' after the removal has a difference of $||Z - Z'|| = ||X * f_{:,1} * g_{1,:}||$ from the original output Z.

Instead of simply removing $f_{:,1}$ and Y_1, if we approximate $f_{:,1}$ using Eq. 1, where the first filter can be expressed as $f_{:,1} = \sum_{l\neq1} \lambda_{1,l} f_{:,l} + \epsilon_1$, we can achieve more improved pruning. By substituting this approximation to Eq. 6, we get

$$Z = \sum_{j\neq1} X * f_{:,j} * (g_{j,:} + \lambda_{1,j} g_{1,:}) + X * \epsilon_1 * g_{1,:}. \quad (7)$$

This shows the fact that the output feature map Z can be expressed without using $f_{:,1}$. After we eliminate the first channel, if we change all the weights $g_{j,:}$ to $g_{j,:}+\lambda_{1,j}g_{1,:}$, then the output feature map Z'' in that situation has a feature map difference of $||Z-Z''|| = ||X*\epsilon_1*g_{1,:}||$ which would be much smaller than the $||Z - Z'||$ in the above removal without weight modification. We call this modification as *Weights Compensation* which implies that the difference caused by pruning can be compensated by direct weight change.

Without loss of generality, the above process can be extended to other filters $f_{:,j}$, not only for $f_{:,1}$. Therefore, the proposed *weights compensation* can be further generally defined as follows. Whenever we remove a specific l-th channel Y_l, we directly modify the remaining weight value $g_{j,:}$ of

1×1 convolution to $g'_{j,:}$ for all $j \neq l$ where

$$g'_{j,:} = g_{j,:} + \lambda_{l,j} g_{l,:}. \tag{8}$$

Then, the difference in the output caused by pruning can be drastically reduced. Eq. 7 also implies another fact that the difference in output after the removal of the l-th channel does not depend on only the approximation error ϵ_l. The output difference of Z'' after weights compensation is $||Z - Z''|| = ||X * \epsilon_l * g_{l,:}||$, which is also a function of the weight g. Therefore, we finally propose a selection criterion that considers both ϵ and g. Intuitively, if both ϵ_l and $g_{l,:}$ are of smaller magnitude, it is more likely to have small $||X * \epsilon_l * g_{l,:}||$. Thus, we replace the previous criterion in Eq. 4 by the following selection criterion, where we remove the s-th channel such that

$$s = \arg\min_{l} ||\epsilon_l|| \cdot ||g_{l,:}||. \tag{9}$$

Our pruning process is summarized in Alg 1. Because we already have a pre-trained network, f, g, λ, ϵ can be easily obtained without much cost.

Following the above process, we can reduce the number of output channels of the given $K \times K$ convolution. In addition, if we add the identity initialized 1×1 convolution at the bottom of the $K \times K$ convolution and apply the same process, then we can also reduce the number of input channels of the $K \times K$ convolution. However, this time, the process should be applied symmetrically because the 1×1 convolution is at the bottom. We have to find the replaceable filters using the transposed filter $f_{i,:} \in \mathbb{R}^{K \times K \times n}$ which is the set of weights connected to the i-th input channel of $K \times K$ convolution. The process is briefly illustrated in Figure 2.

LRF can reduce both the number of input and output channels of any convolution without being affected by the network structure or other modules near the convolution. For example, while existing techniques have made it difficult to reduce the channels connected to the skip-connection in ResNet, the proposed method can reduce the computation of $K \times K$ convolution regardless of the existence of nearby skip-connection. The proposed method can be used for all convolution layers with any kernel sizes. Therefore, one of the main advantages of this technique is that it is generally applicable to any type of CNN architecture.

Because weights compensation plays a very important role in the LRF, it is applied together in all further experiments when using the LRF. Therefore, any future reference to LRF henceforth implies the usage of LRF and weights compensation together.

Number of FLOPs and Params

One might wonder whether it is fine to add two additional 1×1 convolution layers. When the number of channels is n, the FLOPs of the original $K \times K$ convolution is $\propto K^2 n^2$. After the addition of two 1×1 convolutions, the FLOPs become $\propto (K^2 + 2) \cdot n^2$. However, when we prune the channels with ratio p, then the FLOPs become $\propto 2n^2(1-p) + K^2 n^2 (1-p)^2$ which is significantly lower than the original FLOPs. For example, if $p = 0.5$ and $K = 3$, it shows 64% of FLOPs reduction.

Algorithm 1: Iterative algorithm for single layer

Given: Weights $f_{i,j} \in \mathbb{R}^{K \times K}$, $g_{j,k} \in \mathbb{R}$, pruning ratio p, set of removed channel indices P, set of remaining channel of indices T
Initialization: $P \leftarrow \emptyset$, $T \leftarrow \{1, \ldots, n\}$
while $|P| < n \times p$ **do**
  Obtain $\lambda_{:,:}$, $\epsilon_:$ from channels in T using Eq. 1, 3
  Find index s using the Eq. 9, and move s from T to P
  Remove s-th output channel and associated weights
  for *j in T* **do**
    $g_{j,:} \leftarrow g'_{j,:} = g_{j,:} + \lambda_{s,j} g_{s,:}$
  fine-tune $\phi(\cdot)$
Move on to the pruning of input channels of $K \times K$ convolution, and repeat the above process symmetrically
Move on to the next layer

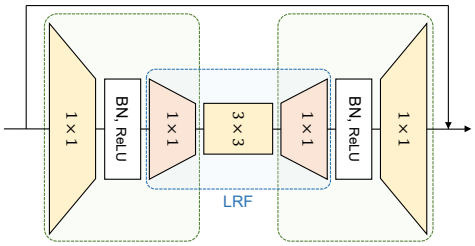

Figure 3: General view of pruning for bottleneck block. Orange convolution layers are newly added for LRF. Channels related to 3×3 convolutions are already pruned using LRF (blue box), and channels between consecutive 1×1 convolutions (green box) will be subsequently pruned with Eq. 10.

Pruning for Bottleneck

Unlike the usual network, the FLOPs of bottleneck block in ResNet are concentrated on the 1×1 convolution layer. Therefore, in our basic framework, it is difficult to achieve a significantly high compression rate in the deeper ResNet. Here, we propose the modified small-norm criterion as an additional method to reduce the FLOPs in bottleneck block.

In the bottleneck block, the 3×3 convolution in the middle can be pruned first using LRF with the addition of two 1×1 convolutions (blue box in Figure 3). Then, two pairs of two consecutive 1×1 convolutions remain at the block as shown in the green boxes of Figure 3. The traditional small-norm criterion applied to 3×3 convolution was a problematic method because that considers only the magnitude of weights and ignores the spatial computation mechanism of convolution. Unlike 3×3 convolution, 1×1 convolution does not observe the spatial relationship in the feature map, therefore the scale of 1×1 convolution contains much significant information. In addition, since the feature map is multiplied by the BN scaling factor γ_k, this is also an important factor to consider when choosing the channel to prune. Using these facts, we propose to use the following modified small-norm criterion to prune the channel between the two 1×1 convolutions. We remove the s-th channel such that

$$s = \arg\min_{k} ||w^{lower}_{:,k}|| \cdot \gamma_k \cdot ||w^{upper}_{k,:}|| \tag{10}$$

Network	Method	Baseline Acc (%)	Pruned Acc (%)	Acc ↓ (%)	FLOPs ↓ (%)	Param ↓ (%)
ResNet-32	SFP (He et al. 2018)	92.63	92.08	0.55	41.5	-
	LFPC (He et al. 2020)	92.63	92.12	0.51	52.6	-
	FPGM (He et al. 2019)	92.63	91.93	0.70	53.2	-
	LRF-50 (Ours)	**92.49**	**92.54**	**-0.05**	**62.0**	**63.3**
	LRF-60 (Ours)	**92.49**	**92.02**	**0.47**	**73.5**	**74.0**
ResNet-56	NISP (Yu et al. 2018)	-	-	0.03	43.6	42.6
	DCP (Zhuang et al. 2018)	93.80	93.81	-0.01	47.1	70.3
	HRank (Lin et al. 2020)	93.26	93.17	0.09	50.0	42.4
	SFP (He et al. 2018)	93.59	92.26	1.33	52.6	-
	FPGM (He et al. 2019)	93.59	93.49	0.10	52.6	-
	LFPC (He et al. 2020)	93.59	93.24	0.35	52.9	-
	GBN (You et al. 2019)	93.10	93.43	-0.33	60.1	52.5
	LRF-50 (Ours)	**93.45**	**93.73**	**-0.28**	**62.4**	**63.4**
	GBN (You et al. 2019)	93.10	93.07	0.03	70.3	66.7
	LRF-60 (Ours)	**93.45**	**93.19**	**0.26**	**73.9**	**74.1**
	HRank (Lin et al. 2020)	93.26	90.72	2.54	74.1	68.1
ResNet-110	SFP (He et al. 2018)	93.68	93.38	0.30	40.8	-
	NISP (Yu et al. 2018)	-	-	0.18	43.8	43.3
	FPGM (He et al. 2019)	93.68	93.85	-0.17	52.3	-
	HRank (Lin et al. 2020)	93.50	93.36	0.14	58.2	59.2
	LFPC (He et al. 2020)	93.68	93.07	0.61	60.3	-
	LRF-50 (Ours)	**93.76**	**94.34**	**-0.58**	**62.6**	**63.5**
	HRank (Lin et al. 2020)	93.50	92.65	0.85	68.6	68.7
	LRF-60 (Ours)	**93.76**	**94.16**	**-0.40**	**74.1**	**74.2**

Table 1: Comparison results of ResNet on CIFAR-10. *Acc* ↓ (the smaller, the better) is the accuracy drop of pruned model compared to the baseline model. FLOPs ↓ and Param ↓ represent the reduction of FLOPs and parameters in percentage, respectively.

where w^{lower} is a set of weights of the lower 1×1 convolution, and w^{upper} is a set of weights of the upper 1×1 convolution. γ_k is the k-th scaling factor of the BN. Three factors in the above equation encompass every weight connected to the k-th channel between two 1×1 convolutions.

Overall Pruning Framework for LRF

The overall process of LRF is demonstrated in Algorithm 1. Given a pre-trained network, we add two 1×1 convolution layers initialized as an identity at the bottom and top of every $K \times K$ convolution layer. Using LRF, we can reduce the number of input and output channels of the $K \times K$ convolution with a ratio p which is pre-defined by the user. Pruning starts from the top layer which is closest to the classifier because pruning the bottom layer first is likely to affect the value of the top layer. After we prune all the layers from top to bottom, we fine-tune the entire network to improve the generalization ability of the pruned network. In the bottleneck block, we prune the channels related to the $K \times K$ convolution of the entire network using LRF first, and the channels between the consecutive 1×1 convolutions are removed using Eq. 10.

Experiments

Training Details. We used two image classification datasets, CIFAR-10 (Krizhevsky, Hinton et al. 2009) and ImageNet (Russakovsky et al. 2015). CIFAR-10 is a 10-class image classification dataset with an image size of 32×32. It contains 50k training images and 10k validation images. For the CIFAR-10, all the training settings follow the settings of ResNet (He et al. 2016a). ImageNet is a large-scale image classification dataset which contains 1.28 million training images and 50k validation images. For the ResNet on ImageNet experiment, we used the pre-trained network officially provided by PyTorch torchvision (Paszke et al. 2019).
LRF Details. At the pruning stage, we fine-tune the network for one epoch each time we prune a layer. After pruning is finished, we finally fine-tune the pruned network. For the CIFAR-10, the learning rate starts from 0.01, and is divided by 10 at the half of the training. The total fine-tuning epochs are 1.5 times longer than the original training schedule. For the final fine-tuning of ImageNet, general training settings are used with a reduced learning rate to one-tenth of the original. Whenever we fine-tune a pruned network, dark knowledge (Hinton, Vinyals, and Dean 2015) ($T = 2$) is used with the original pre-trained network as the teacher network. Because an original model is already given, no additional cost is required. All the results of LRF are average of three runs.

Comparison Results

ResNet on CIFAR-10. CIFAR-10 is the most widely used dataset to evaluate pruning performance. ResNet is the basis of the latest CNN models which show state-of-the-art performance in several computer vision tasks. Since recent networks such as DenseNet (Huang et al. 2017), pre-activated ResNet (He et al. 2016b), and Wide-ResNet (Zagoruyko

Network	Method	Pruned Top-1 (%)	Top-1 ↓ (%)	Pruned Top-5 (%)	Top-5 ↓ (%)	FLOPs ↓ (%)	Param ↓ (%)
ResNet-18	MIL (Dong et al. 2017)	66.33	3.65	86.94	2.30	34.6	-
	SFP (He et al. 2018)	67.10	3.18	87.78	1.85	41.8	-
	FPGM (He et al. 2019)	68.41	1.87	88.48	1.15	41.8	-
	LRF-40 (Ours)	**69.78**	**-0.02**	**89.32**	**-0.25**	**45.5**	**46.9**
	LRF-50 (Ours)	**69.07**	**0.69**	**88.99**	**0.08**	**57.6**	**59.7**
ResNet-50	SFP (He et al. 2018)	74.61	1.54	92.06	0.81	41.8	-
	HRank (Lin et al. 2020)	74.98	1.17	92.33	0.54	43.8	36.7
	NISP (Yu et al. 2018)	-	0.89	-	-	44.0	43.8
	GDP (Lin et al. 2018)	71.89	3.24	90.71	1.59	51.3	-
	LRF-50 (Ours)	**75.78**	**0.43**	**92.85**	**-0.03**	**51.8**	**49.1**
	FPGM (He et al. 2019)	74.83	1.32	92.32	0.55	53.5	-
	GBN-50 (You et al. 2019)	75.18	0.67	92.41	0.26	55.1	53.4
	ThiNet (Luo, Wu, and Lin 2017)	71.01	1.87	90.02	1.12	55.8	51.6
	DCP (Zhuang et al. 2018)	74.95	1.06	92.32	0.51	55.8	51.5
	LRF-60 (Ours)	**75.71**	**0.50**	**92.80**	**0.02**	**56.4**	**53.5**

Table 2: Comparison results of ResNet on ImageNet. *Pruned Top-1* and *Pruned Top-5* denote the top-1 and top-5 accuracy after the pruning. *Top-1* ↓ and *Top-5* ↓ denote the accuracy drop of the pruned model compared to the baseline model.

and Komodakis 2016) are all based on the basic ResNet, the pruning ability on ResNet represents that the proposed method could be generally applicable to modern networks. We conducted experiments on ResNet using three different depths, 32, 56, and 110.

The pruning results are shown in Table 1. LRF-60 represents that we have pruned 60% of the channels. Our LRF outperforms the previous state-of-the-art models on several criteria. All our models have reduced the FLOPs by more than 60% and nonetheless can outperform most existing networks. For the ResNet-32, LRF-50 outperforms the previous SOTA model FPGM although 16% more FLOPs are reduced. For the ResNet-56, LRF-60 also shows the best performance compare to the previous SOTA model GBN and HRank. Compared to GBN, our model achieves a slightly lower accuracy increase, but the comparison is unfair because GBN used a baseline with low accuracy. The final accuracy is actually more important than the accuracy drop when evaluating the performance of the pruning method. To confirm that, we conducted a detailed analysis of the correlation between the baseline accuracy and the final accuracy afterward.

ResNet on ImageNet. Unlike the ResNet on CIFAR which only uses basic blocks, ResNet on ImageNet uses bottleneck blocks for deeper networks. To demonstrate that LRF works for any structures, we used both ResNet-18 which is composed of basic blocks and ResNet-50 which employs the bottleneck blocks. The results are shown in Table 2.

Regarding ResNet-18, LRF-40 achieves the highest performance with a margin of 1.37% than FPGM in Top-1 accuracy. For the first time, on ResNet-18, our pruned model has succeeded in outperforming the original model. In ResNet-50, which has a bottleneck structure, the channels between the 1×1 convolutions have been removed by half. For the ResNet-50, we outperform the previous SOTA in both Top-1 and Top-5 accuracy while achieving more FLOPs reduction. Especially, the accuracy drop of the Top-5 is only 0.02% which represents that the accuracy of the original model is almost maintained.

Model	Training/Inference (ms)	Volume (MB)
Baseline	566.9 / 157.9	100.1
LRF-50	244.4 / 78.3	51.1
LRF-60	226.6 / 77.8	46.5

Table 3: Training/inference time & Volume of the model.

Method	Baseline (%)	Pruned (%)	Scratch	Scratch-long
LRF-50	93.45	**93.73**	89.85	91.44
LRF-60	93.45	**93.19**	89.81	91.13

Table 4: Investigation on the training from scratch at ResNet-56 on CIFAR-10. *scratch* indicates that the training our final pruned network from scratch. *scratch-long* is similar, but training epochs are extended to be fairly compared.

Analyses on LRF

Training/Inference time & Volume of the model. 1) We measured the actual training/inference time of the ResNet-50 on four RTX 2080Ti GPU with a batch size of 256. The results are shown in Table 3. Despite the addition of several 1×1 convolutions, the actual times are improved substantially. This result is due to the fact that the amount of computation is more important to the training speed than the effect from the number of layers. 2) Actual volume of the ResNet-50 models are also shown in Table 3. These results resolve the question about the practical advantage of our method.

Training from Scratch. LRF is a pruning method which utilizes the properties of 1x1 convolution when removing existing convolution layers. And, it is quite different from designing a new efficient network structure. However, one might suspect that the effectiveness of our work could come

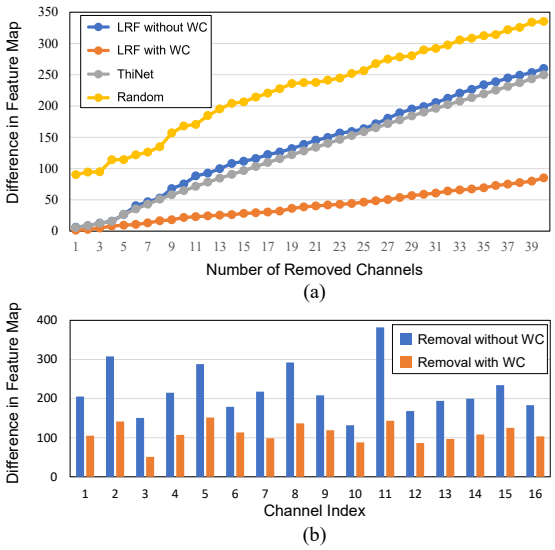

Figure 4: (a) Difference in output feature map that occurs when removing channels iteratively following each method. LRF with weights compensation shows significantly better results than other methods. (b) Difference in output feature maps when removing each channel in a layer. The horizontal axis implies the index of channel that we removed. Each 16 channel shows a significant difference.

from the addition of 1x1 convolution in architecture.

We trained our newly changed final model after pruning from the scratch. To compare them fairly, we also trained 'scratch-long' which has the same total training epochs with our LRF pruning. The comparison result is shown in Table 4. This result proves that our benefit comes from the suggested LRF technique, not from the change in structure.

Effectiveness of Weights Compensation. LRF with weights compensation can reduce the differences in output feature map caused by pruning, and it enables the pruned network to preserve its original performance. To demonstrate the effectiveness of our proposed method, we observed the difference in output feature maps when pruning each channel based on each baseline method.

Given a $K \times K$ convolution layer that we try to prune, we first record the output feature map of the next layer for a particular training input batch. Then, we remove the most redundant channel one by one iteratively following each method, and calculate the norm of the difference between the resulting output feature map and the already-recorded value. There are four baselines, namely, 1) *Random*: choose the channel randomly, 2) *ThiNet*: greedily find the channel with the smallest difference, 3) *LRF without weights compensation*: apply LRF, but simply remove the channel without weight modification, 4) *LRF with weights compensation*: use LRF and modify the weight values with the proposed weights compensation method. For this ablation, the last convolution layer of ResNet-56 on CIFAR-10 was used.

The results are shown in Figure 4a. Using LRF without weights compensation shows a slightly larger difference than that of ThiNet, which is a greedy way of finding the smallest difference. However, LRF with weights compensation shows outstanding results with a much lower difference in output feature maps.

Baseline Acc (%)	Pruned Acc (%)	Acc Drop (%)
92.91	93.13 ($\pm$ 0.14)	-0.22
93.23	93.18 ($\pm$ 0.13)	0.05
93.45	93.27 ($\pm$ 0.11)	0.18
93.55	93.16 ($\pm$ 0.15)	0.39

Table 5: Investigation on the correlation between baseline accuracy and accuracy drop. Although the same pruning method is used for the same network, the pruned accuracy is similar regardless of the baseline accuracy.

In addition to the above ablation, we also observed the distribution of output feature map difference depending on which channel is removed (Figure 4b). In this experiment, we did not remove multiple channels iteratively, but rather recorded the difference after removing each channel independently. There are total 16 bars in the figure because this convolution layer has 16 channels. For all the channels, the value of the orange bar is much smaller than the value of the blue bar. This also shows that weights compensation is good at reducing the output feature map difference. One thing to be noted here is that, the channel selection criterion of ThiNet and LRF are also different. In this result, the ThiNet selects the 10-th channel which has the lowest blue bar. However, we select the third channel that has the lowest orange bar after the weights compensation.

Accuracy Drop vs Baseline Accuracy. Unfortunately, in the field of network pruning, the criterion for comparing each proposed method is unclear. The most standard way is to measure the accuracy drop when the reduced FLOPs are similar. However, the question is whether it is fair to consider only the accuracy drop without considering the baseline accuracy or not. To investigate the actual correlation between the baseline accuracy and the accuracy drop, we observed the performance by applying the same LRF to four baseline models with different performances. As can be seen in Table 5, the performance of each baseline was different, but we ultimately obtained the similar final pruned accuracy. Using a baseline with low performance tends to be an easy way to obtain a good accuracy drop. Therefore, it is unreasonable to evaluate the ability of the pruning method by only comparing the accuracy drop. A better way is to compare the pruned accuracy, rather than the accuracy drop.

Conclusion

In this study, we propose a new effective channel pruning method called Linearly Replaceable Filters (LRF). LRF suggests a novel pruning criterion that selects the channel by using a linear combination approximation of other filters in the same layer. In addition, a technique called weights compensation is proposed to support LRF in various manners. When used together, they significantly reduce output feature map differences in general CNNs regardless of kernel size, block type, and even architectures. Extensive experiments and analyses demonstrate the effectiveness of our approach.

Acknowledgements

This work was supported by the National Research Council of Science & Technology (NST) grant by the Korea government (MSIT) (No. CRC-15-05-ETRI).

References

Dong, X.; Huang, J.; Yang, Y.; and Yan, S. 2017. More is less: A more complicated network with less inference complexity. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 5840–5848.

Han, S.; Pool, J.; Tran, J.; and Dally, W. 2015. Learning both weights and connections for efficient neural network. In *Advances in neural information processing systems*, 1135–1143.

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

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2016b. Identity mappings in deep residual networks. In *European conference on computer vision*, 630–645. Springer.

He, Y.; Ding, Y.; Liu, P.; Zhu, L.; Zhang, H.; and Yang, Y. 2020. Learning Filter Pruning Criteria for Deep Convolutional Neural Networks Acceleration. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2009–2018.

He, Y.; Kang, G.; Dong, X.; Fu, Y.; and Yang, Y. 2018. Soft filter pruning for accelerating deep convolutional neural networks. *arXiv preprint arXiv:1808.06866* .

He, Y.; Liu, P.; Wang, Z.; Hu, Z.; and Yang, Y. 2019. Filter pruning via geometric median for deep convolutional neural networks acceleration. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 4340–4349.

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.

Hinton, G.; Vinyals, O.; and Dean, J. 2015. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531* .

Howard, A. G.; Zhu, M.; Chen, B.; Kalenichenko, D.; Wang, W.; Weyand, T.; Andreetto, M.; and Adam, H. 2017. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861* .

Hu, H.; Peng, R.; Tai, Y.-W.; and Tang, C.-K. 2016. Network trimming: A data-driven neuron pruning approach towards efficient deep architectures. *arXiv preprint arXiv:1607.03250* .

Huang, G.; Liu, Z.; Van Der Maaten, L.; and Weinberger, K. Q. 2017. Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 4700–4708.

Ioffe, S.; and Szegedy, C. 2015. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167* .

Krizhevsky, A.; Hinton, G.; et al. 2009. Learning multiple layers of features from tiny images. Technical report, Citeseer.

Krizhevsky, A.; Sutskever, I.; and Hinton, G. E. 2012. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, 1097–1105.

Lin, M.; Ji, R.; Wang, Y.; Zhang, Y.; Zhang, B.; Tian, Y.; and Shao, L. 2020. HRank: Filter Pruning using High-Rank Feature Map. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 1529–1538.

Lin, S.; Ji, R.; Li, Y.; Wu, Y.; Huang, F.; and Zhang, B. 2018. Accelerating Convolutional Networks via Global & Dynamic Filter Pruning. In *IJCAI*, 2425–2432.

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.; Mu, H.; Zhang, X.; Guo, Z.; Yang, X.; Cheng, T. K.-T.; and Sun, J. 2019. MetaPruning: Meta Learning for Automatic Neural Network Channel Pruning. *arXiv preprint arXiv:1903.10258* .

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

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.

Paszke, A.; Gross, S.; Massa, F.; Lerer, A.; Bradbury, J.; Chanan, G.; Killeen, T.; Lin, Z.; Gimelshein, N.; Antiga, L.; Desmaison, A.; Kopf, A.; Yang, E.; DeVito, Z.; Raison, M.; Tejani, A.; Chilamkurthy, S.; Steiner, B.; Fang, L.; Bai, J.; and Chintala, S. 2019. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In *Advances in Neural Information Processing Systems 32*, 8024–8035.

Romero, A.; Ballas, N.; Kahou, S. E.; Chassang, A.; Gatta, C.; and Bengio, Y. 2014. Fitnets: Hints for thin deep nets. *arXiv preprint arXiv:1412.6550* .

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. doi:10.1007/s11263-015-0816-y.

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

Tan, M.; and Le, Q. V. 2019. EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks. *arXiv preprint arXiv:1905.11946* .

Xie, S.; Girshick, R.; Dollár, P.; Tu, Z.; and He, K. 2017. Aggregated residual transformations for deep neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 1492–1500.

Yim, J.; Joo, D.; Bae, J.; and Kim, J. 2017. A gift from knowledge distillation: Fast optimization, network minimization and transfer learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 4133–4141.

You, Z.; Yan, K.; Ye, J.; Ma, M.; and Wang, P. 2019. Gate decorator: Global filter pruning method for accelerating deep convolutional neural networks. *arXiv preprint arXiv:1909.08174* .

Yu, R.; Li, A.; Chen, C.-F.; Lai, J.-H.; Morariu, V. I.; Han, X.; Gao, M.; Lin, C.-Y.; and Davis, L. S. 2018. Nisp: Pruning networks using neuron importance score propagation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 9194–9203.

Zagoruyko, S.; and Komodakis, N. 2016. Wide residual networks. *arXiv preprint arXiv:1605.07146* .

Zhang, X.; Zhou, X.; Lin, M.; and Sun, J. 2018. Shufflenet: An extremely efficient convolutional neural network for mobile devices. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 6848–6856.

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*, 875–886.