Linear Combination Approximation of Feature for Channel Pruning

Donggyu Joo   Doyeon Kim   Eojindl Yi   Junmo Kim
Korea Advanced Institute of Science and Technology (KAIST)
{jdg105, doyeon_kim, djwld93, junmo.kim}@kaist.ac.kr

Abstract

Network pruning is an effective method that reduces the computation of neural networks while maintaining high performance. This enables the operation of deep neural networks in resource-limited environments. In a general large network, the roles of each channel often inevitably overlap with those of others. Therefore, for more effective pruning, it is important to observe the correlation between features in the network. In this paper, we propose a novel channel pruning method, namely, the linear combination approximation of features (LCAF). We approximate each feature map by a linear combination of other feature maps in the same layer, and then remove the most approximated one. Additionally, by exploiting the linearity of the convolution operation, we propose a supporting method called weight modification, to further reduce the loss change that occurs during pruning. Extensive experiments show that LCAF achieves state-of-the-art performance in several benchmarks. Furthermore, ablations on the LCAF demonstrate the effectiveness of our approach in a variety of ways.

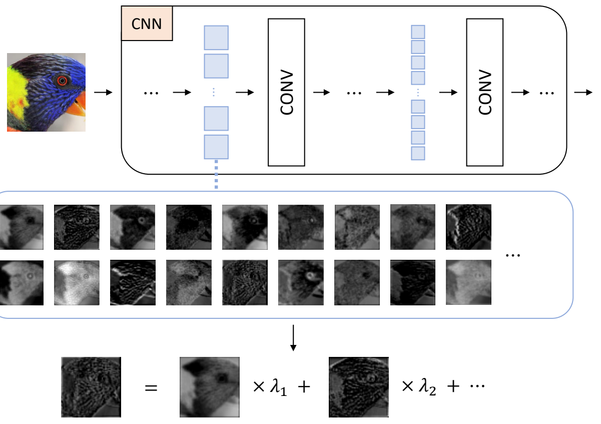

Figure 1. Graphical view of feature maps between the convolutional layers in the CNN. We approximate each feature map by a linear combination of other features. The feature maps are visually similar because they share the same spatial information. At deeper layers, the approximation becomes easier because the dimensions decrease and the number of channels increases.

1. Introduction

With the advent of convolutional neural networks (CNNs), significant performance improvements have been achieved in many computer vision research areas [4, 13, 17, 35]. As the number of available datasets increases and computational resources become more powerful, many deep learning models require higher computing costs and memory footprints for inference. However, considering that many real-world tasks are performed in real-time, demands for heavy computation and enormous model size limit the applicability of deep neural networks. Therefore, there remains a need for an efficient approach that can resolve the problem of computational costs while preserving accuracy.

With a given pre-trained network, network pruning aims to create a more light network that can achieve accurate estimation and efficient inference time. Early Pruning research [3] has primarily focused on removing individual weights that have a minor impact on the performance of deep neural networks. However, this weight pruning method has difficulty in speeding up the CNN in practice because of its unstructured sparsity. Subsequently, channel pruning methods that eliminate each channel with all their connections are more preferred because of their structural and practical advantages [8, 24, 41]. In this study, we also adopt channel pruning to take these benefits.

Recent pruning studies can be categorized into two directions, which are redundancy-based pruning [3, 24, 27], and replaceability-based pruning [8, 20]. *Redundancy*, a traditional well-known concept in pruning, focuses on a specific filter which has the least influence on the computation of the network. The redundancy-based method removes the filter with the lowest impact without considering the upcoming fine-tuning stage, *e.g.* filter with the smallest norm. When this filter is removed, the network is compressed without inducing much difference. However, this approach has the drawback that it neglects the existence of a fine-tuning stage. In general, after pruning is completed, the remaining parameters are re-trained to recover the network. Therefore, less impact at the moment of pruning does not guarantee high performance at the final. On the other hand, the *replaceability*-based approach [8, 20] for pruning concentrates on the correlation between features while also taking into account the re-training stage that follows the pruning. When it removes a replaceable filter, the loss might increase at that instant. However, it can soon be recovered to a better optimal point during the fine-tuning stage if the eliminated filter has replaceability. However, this approach also has a disadvantage in that there is often a lack of clear theoretical explanation to support its reasonable intuition.

To take the strengths of both methods, we propose a novel channel pruning method called *linear combination approximation of features* (LCAF), which prunes the channel by considering the correlation between features. In the LCAF, we approximate each feature map as a linear combination of other feature maps in the same layer (Figure 1). Then, the feature with the lowest approximation error is regarded as a replaceable feature to be pruned. After pruning, the remaining features in the same layer successfully cover the role of the removed one. The LCAF framework is composed of an additional crucial method *weight modification* to support the theoretical basis of LCAF. By exploiting the linearity of the convolutional layer, the loss increase of the network can be further reduced when we directly modify the remaining weights to appropriate values. This method takes advantage of the redundancy-based approach, which minimizes the actual change in the network.

A graphical view of a CNN in Figure 1 shows the feature maps at the middle of a pre-trained network. The main idea of LCAF is inspired by the following observations on these features. First, the feature maps in the same layer are visually similar. More precisely, they are spatially well-aligned owing to the properties of the CNN operation. Information from the same receptive field exists at exactly the same location for every feature. Therefore, the features can be expressed with others by a simple linear operation. Second, in the deeper layers, the size of the feature map decreases, and the number of channels increases. Theoretically, the span of vectors is more likely to cover all spaces if there are many vectors with low dimensions. Apart from their visual similarity, this mathematical perspective implies that the features can be successfully expressed by a combination of other features. Third, with the concept of replaceability, the role of the linearly approximated feature can be recovered after the pruning when their roles overlap a lot.

LCAF is not only an intuitive idea but has also shown remarkable results in extensive experiments. We achieved state-of-the-art performance in CIFAR-10 [16] and ImageNet [33] experiments. Various ablation studies have also proven the efficacy of LCAF in many respects.

Our contributions can be summarized as follows:

- We propose a novel channel pruning method, LCAF, which prunes the most approximated feature by the linear combination of other features.
- The loss change after pruning can be further reduced by directly modifying the remaining weights in convolutional layers to appropriate values. We call this a *weight modification* method.

2. Related Works

Network Pruning. Pruning is a technique that removes unnecessary parameters of the network to make a compact and fast model. As previously described, early pruning began with a weight-removal approach [3]. However, because each channel of CNNs is related to a large number of weights, pruning of each weight results in an unstructured network. Subsequently, this makes it difficult to utilize the network in a BLAS library, and it does not result in a faster inference of the CNNs practically. In contrast, channel pruning, which eliminates individual channels with all connected weights, accelerates the inference time because entire channels are eliminated. Early channel pruning studies used several heuristic approaches, such as Lasso regression [9, 27], or removing zero-activation neurons in the network [12]. Zhuang *et al.* [41] proposed discriminative-aware channel pruning and discrimination-aware losses to select the most significant channels for the networks. He *et al.* [8] focused on eliminating filters close to the geometric median of the filters within a layer. LRF [15] proposed the linearly replaceable filter which utilizes the linear approximation between the filter values. LRF shows notable results in terms of FLOPs, but it requires a number of additional layers which damage the inference time. Furthermore, in recent years, some novel methods have adopted meta-learning [19, 25] or reinforcement learning [7].

Network pruning has a variety of distinct directions, unlike the traditional way which compresses the pre-trained network. Liu *et al.,* [26] doubts the benefit of the pruning framework by comparing it to the scratch training with extended training epochs. Li *et al.,* [18] and Frankle *et al.,* [1] try to obtain the optimal sub-network structure using the characteristics of the large network, and then train it from the scratch, not compressing from the pre-trained network.

Other Methods. In addition to pruning, various approaches are being conducted to improve the efficiency of the network. The knowledge distillation technique [10, 32, 38] extracts information from a large pre-trained network and transfers it to a small network. Therefore, the network exhibits reasonable performance with fewer parameters. The network architecture search (NAS) method [23, 31, 42] automatically finds the network structure instead of hand-designed architecture. It is similar to pruning in that a model

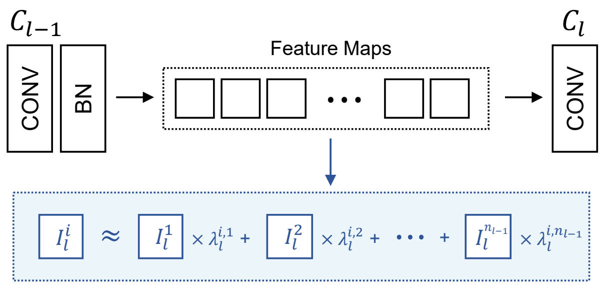
C_{l-1}
CONV
BN
Feature Maps
C_l
CONV
$I_l^i \approx I_l^1 \times \lambda_l^{i,1} + I_l^2 \times \lambda_l^{i,2} + \cdots + I_l^{n_{l-1}} \times \lambda_l^{i,n_{l-1}}$

Figure 2. Feature maps between two convolutional layers. Each feature map can be approximated as a linear combination of the other feature maps since they have the same dimensions. The following C_l is further modified by weight modification to reduce the loss change.

can employ fewer parameters to achieve a certain performance while finding a proper structure for a task. In addition, in some studies, networks have been manually designed to achieve faster inference of the model [2,11,37,40]. Because these methods are complementary to ours, they can be further combined with our proposed approach.

3. Method

3.1. Notations

A pre-trained network $\phi(\cdot)$ has a set of N convolutional layers $\{C_1, C_2, \cdots C_N\}$, where C_l denotes the l-th convolutional layer. C_l has a weight $\mathcal{W}_l \in \mathbb{R}^{n_{l-1}\times n_l \times k_l \times k_l}$, where n_l is the number of output channels of C_l, and $k_l \times k_l$ is the kernel size. For convenience, we denote the collection of weights connected to the q-th output channel as $\mathcal{W}_l^{:,q}$ and the collection of weights connected to the p-th input channel as $\mathcal{W}_l^{p,:}$. The input feature maps of C_l are denoted by $\mathcal{I}_l = \{I_l^1, I_l^2, \cdots, I_l^{n_{l-1}}\} \in \mathbb{R}^{b\times n_{l-1}\times h_l \times w_l}$, where b is the batch size of the input, and h_l and w_l are the height and width, respectively. The i-th input feature map I_l^i is connected to $\mathcal{W}_l^{i,:}$ of C_l to generate the output feature map. The convolution operation of C_l can then be defined as follows:

$$C_l(\mathcal{I}_l) = \sum_{k=1}^{n_{l-1}} I_l^k * \mathcal{W}_l^{k,:} \tag{1}$$

where $*$ denotes the 2D convolution operation.

3.2. Linear Approximation of Features

For the input feature $\mathcal{I}_l = \{I_l^1, I_l^2, \cdots, I_l^{n_{l-1}}\}$ from any layer, each i-th feature map has the same dimension as the others. Therefore, as shown in Figure 2, any I_l^i can be expressed as a linear combination of the other features as follows:

$$I_l^i = \sum_{k\neq i} \lambda_l^{i,k} I_l^k + \epsilon_l^i \tag{2}$$

where $i \in \{1, 2, \cdots, n\}$, $\lambda_l^{i,k}$ are scalar coefficients, and ϵ_l^i is an approximation error that has the same dimensions as I_l^i. As shown in Figure 1, these features share many characteristics; therefore, this linear combination yields a reasonable approximation.

In CNN, the role of each feature is closely related to how the feature is correlated with other features. Therefore, we regard the most linearly approximated feature as the channel to be pruned. The role of this feature can be successfully covered by the remaining features since the fine-tuning process typically follows immediately after each pruning step to recover the damaged network in the pruning field.

Our goal is to approximate each feature as best as possible by a linear combination of other features. Each $\lambda_l^{i,k}$ can be obtained from the following optimization problem:

$$\arg\min_{\lambda_l^{i,k}} ||\epsilon_l^i|| = \arg\min_{\lambda_l^{i,k}} ||I_l^i - \sum_{k\neq i} \lambda_l^{i,k} I_l^k|| \tag{3}$$

To simplify the problem, we flatten each tensor I to a column vector $\tilde{I}$. Using partial derivatives with respect to $\lambda_l^{i,1}$, we obtain

$$\frac{\partial ||\epsilon_l^i||^2}{\partial \lambda_l^{i,1}} = -2\tilde{I}_l^{1\top}\tilde{I}_l^i + 2\tilde{I}_l^{1\top}(\sum_{k\neq i} \lambda_l^{i,k}\tilde{I}_l^k) = 0 \tag{4}$$

$$\sum_{k\neq i} \tilde{I}_l^{1\top}\tilde{I}_l^k \lambda_l^{i,k} = \tilde{I}_l^{1\top}\tilde{I}_l^i \tag{5}$$

Without loss of generality, we obtain an $n_{l-1} - 1$ system of linear equations from the partial derivative with respect to each $\lambda_l^{i,k}$. Then, this system of linear equations can be organized as the following simple matrix computation:

$$A^\top A \Lambda = A^\top \tilde{I}_l^i \tag{6}$$

where $A = \begin{bmatrix}\tilde{I}_l^1 & \tilde{I}_l^2 & \cdots & \tilde{I}_l^{i-1} & \tilde{I}_l^{i+1} & \cdots & \tilde{I}_l^{n_{l-1}}\end{bmatrix}$ and $\Lambda = \begin{bmatrix}\lambda_l^{i,1} & \lambda_l^{i,2} & \cdots & \lambda_l^{i,i-1} & \lambda_l^{i,i+1} & \cdots & \lambda_l^{i,n_{l-1}}\end{bmatrix}^\top$.

After obtaining all $\lambda_l^{i,k}$, each ϵ_l^i is computed from Equation 2. This process does not require fine-tuning or back-propagation which incurs a heavy computational cost. The detailed time complexity of this calculation is described in Section 5.4. A more detailed mathematical derivation of this process can be found in the supplementary material.

3.3. Loss Reduction by Weight Modification

The convolutional layer C_l that follows immediately after the input feature map $\mathcal{I}_l$ has linearity. This makes an additional useful method available. After the removal of a channel, the loss of the network can be further reduced by directly modifying the value of $\mathcal{W}_l$ to the appropriate value.

The aim of our channel pruning is to remove the i-th feature map I_l^i and all its connected weights $\mathcal{W}_l^{i,:}$. We assume that the specific i-th feature map I_l^i can be approximated as $I_l^i = \sum_{k \neq i} \lambda_l^{i,k} I_l^k + \epsilon_l^i$ from Equation 2. By substituting this into Equation 1, the operation of C_l can be approximated without using the i-th feature and $\mathcal{W}_l^{i,:}$ as follows:

$$\begin{aligned} C_l(\mathcal{I}_l) &= \sum_{k \neq i} I_l^k * \mathcal{W}_l^{k,:} + (\sum_{k \neq i} \lambda_l^{i,k} I_l^k + \epsilon_l^i) * \mathcal{W}_l^{i,:} \\ &= \sum_{k \neq i} I_l^k * (\mathcal{W}_l^{k,:} + \lambda_l^{i,k} \mathcal{W}_l^{i,:}) + \epsilon_l^i * \mathcal{W}_l^{i,:} \\ &\approx \sum_{k \neq i} I_l^k * (\mathcal{W}_l^{k,:} + \lambda_l^{i,k} \mathcal{W}_l^{i,:}) \end{aligned} \tag{7}$$

Since ϵ_l^i is an approximation error with a low norm, the $C_l(\mathcal{I}_l)$ can be approximated as above. The last line of Equation 7 seems to be another form of the simple 2D convolution operation of $n_{l-1} - 1$ input channels without I_l^i. If we appropriately modify the weight values of C_l, this derivation can be further expressed as follows:

$$C_l(\mathcal{I}_l) \approx \sum_{k \neq i} I_l^k * (\mathcal{W}_l^{k,:} + \lambda_l^{i,k} \mathcal{W}_l^{i,:}) = \overline{C}_l(\mathcal{I}_l^{-i}), \tag{8}$$

where we define $\mathcal{I}_l^{-i} = \{I_l^1, \cdots, I_l^{i-1}, I_l^{i+1}, \cdots, I_l^{n_{l-1}}\}$, which does not include the i-th feature map. And $\overline{C}_l$ is defined as a modified version of C_l with having a weight $\overline{\mathcal{W}}_l \in \mathbb{R}^{(n_{l-1}-1) \times n_l \times k_l \times k_l}$, where

$$\overline{\mathcal{W}}_l^{k,:} = \mathcal{W}_l^{k,:} + \lambda_l^{i,k} \mathcal{W}_l^{i,:}. \tag{9}$$

for all $k \neq i$.

The above derivations imply the following statement: When we remove a certain channel, the change in loss will be further reduced if we modify the remaining weights to appropriate values. In this work, we modify the remaining weight values as shown in Equation 9 whenever we prune a channel. This procedure is referred to as the *weight modification*. The conventional pruning method simply removes a channel that is considered to be unnecessary. However, the loss increase always exists during pruning since the value of the feature map is not zero. Compared to existing studies, weight modification is always effective in that it reduces the loss change.

In existing works, pruning the i-th channel and all its connected weights is equivalent to a situation in which a feature map I_l^i becomes a zero matrix. However, owing to the weight modification, our pruning process is equivalent to converting I_l^i to $I_l^i - \epsilon_l^i$ from the following formulation:

$$\begin{aligned} \overline{C}_l(\mathcal{I}_l^{-i}) &= C_l(\mathcal{I}_l) - \epsilon_l^i * \mathcal{W}_l^{i,:} \\ &= C_l(\{I_l^1, \cdots, I_l^{i-1}, I_l^i - \epsilon_l^i, I_l^{i+1}, \cdots, I_l^{n_{l-1}}\}) \end{aligned} \tag{10}$$

In the LCAF, owing to the benefits of weight modification, the pruning of the channel is equivalent to converting I^i to $I^i - \epsilon^i$. This is a significantly smaller change compared to conventional pruning, which makes I^i zero. Subsequently, our problem becomes measuring the importance of each ϵ_l^i, or how the network is affected when each feature value is reduced by an amount ϵ^i.

3.4. Criterion for Global Pruning

Each i-th feature map in each layer can be linearly approximated using Equation 2. Further loss reduction is possible owing to weight modification. After that, to prune a model, we need to set a channel selection criterion that determines the most replaceable channel. In this study, a global pruning criterion, rather than layer-wise pruning, is employed. We propose two pruning criteria: *normalized norm* and *gradient-based*.

Normalized Norm. As described above, LCAF pruning is equivalent to changing I_l^i to $I_l^i - \epsilon_l^i$. Since the difference caused by pruning is ϵ_l^i, only the value of ϵ_l^i needs to be considered, regardless of the value of I_l^i. From this observation, we propose a *normalized norm* criterion as follows:

$$\arg\min_{l,i} \frac{||\epsilon_l^i||}{\sum_k ||\epsilon_l^k||} \tag{11}$$

The denominator $\sum_k ||\epsilon_l^k||$ is adopted as the normalization for global pruning. Without this normalization, pruning mainly occurs in the deeper layers since the feature map tends to have a smaller norm (smaller feature map size) in a deeper layer. The ablation on the existence of this normalization is presented in Section 5.3.

Gradient-Based Approach. Estimation of the loss change using the gradient is known to be an effective approach [28, 39]. To compare this approach with our normalized norm criterion, we also establish the gradient-based criterion as follows. In our LCAF scheme, the loss difference $\Delta\mathcal{L}$ after pruning is a function $w.r.t.$ I_l^i.

$$|\Delta\mathcal{L}(I_l^i)| = |\mathcal{L}(I_l^i) - \mathcal{L}(I_l^i - \epsilon_l^i)| \tag{12}$$

Inspired by Molchanov *et al.* [28], we use the Taylor series to expand $\mathcal{L}(I_l^i - \epsilon_l^i)$. By applying the first-order approximation of the Taylor series, Equation 12 becomes

$$|\Delta\mathcal{L}(I_l^i)| \approx |\epsilon_l^i \cdot \frac{\partial \mathcal{L}}{\partial I_l^i}| \tag{13}$$

where $\cdot$ is an inner product, and $\partial\mathcal{L}/\partial I_l^i$ is a partial derivative of $\mathcal{L}$ with respect to each element in I_l^i. Therefore, this equation indicates that the Taylor series approximation is summed over all elements in I_l^i. To minimize the loss change, the gradient-based criterion is defined to find the channel with the smallest value in Equation 13. In Section 5.3, we reveal this gradient-based criterion could be problematic in real scenarios. The detailed derivation is described in the supplementary material.

3.5. Entire Pruning Process

Given a pre-trained network, the following four processes are repeated until the desired compression rate is reached. 1) m samples of training images are forwarded to obtain the feature map values I_l^i of each layer. 2) Following Equation 11, calculate the normalized norm criterion for every channel. 3) Globally prune k channels with the least importance, and the appropriate weight modification is applied. 4) One epoch of fine-tuning proceeds after every r times of pruning.

After completing the above pruning process, we proceed to the final fine-tuning stage to boost up the performance and improve the generalization ability.

4. Experiments

4.1. Experimental Settings

Datasets. To validate the effectiveness of the proposed method, we used two image classification benchmark datasets: CIFAR-10 [16] and ImageNet [33]. CIFAR-10 is a 10-class classification dataset that contains 50k training images and 10k validation images. ImageNet is a large-scale image classification dataset with 1.2M training images and 50k validation images of 1,000 classes.

Training and Pruning settings. For CIFAR, We followed the original ResNet [4] training settings to prepare a baseline network. For ImageNet, we used the official pretrained model from PyTorch [29]. Standard data augmentation techniques are adopted for all the datasets. The experiments are conducted on four RTX 2080Ti GPUs.

The LCAF simultaneously prunes $k = 10$ channels globally. One epoch of fine-tuning follows the $r = 5$ pruning steps; $m = 256$ samples of training images are used to obtain the feature values. After the pruning process is completed, we re-train the pruned network to increase the generalization ability and boost the performance. The final fine-tuning stage is similar to the original training settings, with a reduced learning rate of one-tenth of the original. We adopt the normalized norm criterion (Eq. 11) for every quantitative evaluation except for the ablation in Section 5.3. All results of LCAF are the average of five runs.

To use the weight modification, we need a convolutional layer after the feature map. For the plain network, every input feature map of the convolutional layer is used as the pruning subject. For ResNet-based architectures that have shortcuts, we prune only the channels inside the block. The input tensor of the first convolutional layer of each block is not used for the LCAF criterion calculation and pruning.

Table 1. Pruning results of ResNet on the CIFAR-10 dataset. *Top-1* is the accuracy after the pruning. *FLOPs* ↓ and *Params* ↓ represents the reduction of FLOPs and parameters in percentage, respectively. The results are ordered by FLOPs ↓ (higher is more compressed).

Method	Top-1 (%)	FLOPs ↓ (%)	Params ↓ (%)
	ResNet-32		
Baseline	92.73	-	-
SFP [6]	92.08	41.5	-
FPGM [8]	92.31	41.5	-
LFPC [5]	92.12	52.6	-
LCAF	**92.44**	**55.4**	**43.9**
	ResNet-56		
Baseline	93.44	-	-
CCP [30]	93.69	47.0	-
DCP [41]	93.81	47.1	70.3
LCAF	**93.91**	**48.0**	**49.2**
HRank [20]	93.17	50.0	42.4
FPGM [8]	93.49	52.6	-
LFPC [5]	93.24	52.9	-
LCAF	**93.45**	**60.0**	**53.1**
DHP [19]	92.94	60.9	58.9
HRank [20]	90.72	74.1	68.1
LCAF	**92.17**	**75.3**	**77.3**
	ResNet-110		
Baseline	93.77	-	-
GAL [22]	92.74	48.5	44.8
HRank [20]	93.36	58.2	59.2
LCAF	**93.92**	**60.1**	**59.8**
LFPC [5]	93.07	60.3	-
HRank [20]	92.65	68.6	68.7
DHP [19]	92.39	78.4	77.6
LCAF	**93.09**	**80.1**	**82.3**

4.2. Comparative Experiments

4.2.1 Results on CIFAR-10

ResNet. We conducted experiments for ResNet [4] on CIFAR-10 at three different depths: 32, 56, and 110. The results of the CIFAR-10 experiment are listed in Table 1. We achieved state-of-the-art performance in all settings. For each network, several different LCAF models are created for fair comparisons with the other methods at similar FLOPs reduction. Since each result has a different compression rate, so it is better to observe the result carefully by considering both accuracy and FLOPs. For ResNet-32, the

Table 2. Comparison results of VGG-16 on CIFAR-10 dataset.

Method	Top-1 (%)	FLOPs ↓ (%)	Params ↓ (%)
Baseline	93.94	-	-
SSS [14]	93.40	3.4	64.0
GAL [22]	90.73	45.2	82.2
HRank [20]	93.43	53.5	82.9
LCAF	**93.52**	**60.1**	**76.6**
HRank [20]	91.23	76.5	92.0
LCAF	**93.06**	**80.1**	**90.6**

LCAF clearly outperforms all previous works by achieving the highest accuracy while reducing more FLOPs. Compared to the LFPC [5], there is a 0.32% improvement in Top-1 accuracy, with a reduction of 3% more FLOPs. For ResNet-56, at an acceleration of approximately 60%, the LCAF is ahead of the others, and even reduces the FLOPs by 8% compared to LFPC [5] and FPGM [8]. At an acceleration of approximately 75%, a considerable margin of 1.45% compared to the HRank [20] is obtained. In the ResNet-110 experiment, the difference in performance between the LCAF and the other methods is more noticeable. LCAF outperforms the others with FLOPs reductions of approximately 60% and 80%. In particular, compared to LFPC, a 20% more reduction in FLOPs is achieved while recording the same performance.

VGG. To validate our effectiveness in a plain network, experiments with the VGG-16 [36] network are conducted. Table 2 shows a performance comparison with other studies. Compared with HRank, LCAF (60% FLOPs reduction) achieves a 0.09% improvement in Top-1 accuracy while reducing 7% more FLOPs. Moreover, LCAF (80% FLOPs reduction) outperforms HRank (76.5% FLOPs reduction) by a much higher margin of 1.83%. Unlike the HRank, LCAF is more robust at a higher compression rate. LCAF proves its pruning ability in both ResNet and plain structure.

MobileNet-V2. The effect of the LCAF is also tested at the MobileNet-V2 [34] on CIFAR-10. Since it is originally designed for ImageNet, to run the MobileNet-V2 on CIFAR-10 (image size of $32 \times 32 \times 3$), we used the modified version. Some of the downsampling layers (1st, 3rd, and 4th) are disabled in this modified version, *i.e* stride of a few layers are changed from 2 to 1. LCAF successfully reduced the FLOPs by 55% by achieving final top-1 accuracy of 94.96% given a baseline of 94.51%.

4.2.2 Results on ImageNet

Experiments on ImageNet were conducted to validate its effectiveness on a large-scale dataset. The ResNet-50 model was used for this experiment, and the results are shown in Table 3. We obtain an accuracy drop of only 0.48% in the Top-5 accuracy, with over 60% acceleration. Among the existing studies, two methods, LFPC [5] and HRank [20] reduced FLOPs by more than 60%. While reducing a similar percentage of FLOPs, we outperformed them in terms of both Top-1 accuracy and Top-5 accuracy. Specifically, we recorded a 3.16% improvement in Top-1 accuracy and 1.37% in Top-5 accuracy compared to HRank while achieving the same FLOPs reduction. This margin is a significant difference in this field, which proves the effectiveness of our proposed method. Compared to the LFPC, LCAF also outperforms it in all aspects, namely, Top-1 accuracy, Top-5 accuracy, and FLOPs reduction. The performance improvement of LCAF in both small-scale and large-scale benchmarks describes the efficacy of LCAF in general settings.

Table 3. Comparison results of ResNet-50 on ImageNet dataset.

Method	Top-1 (%)	Top-5 (%)	FLOPs ↓ (%)	Param ↓ (%)
Baseline	76.13	92.86	-	-
SFP [6]	74.61	92.06	41.8	-
HRank [20]	74.98	92.33	43.8	36.7
GDP [21]	71.89	90.71	51.3	-
FPGM [8]	74.83	92.32	53.5	-
ThiNet [27]	71.01	90.02	55.8	51.6
DCP [41]	74.95	92.32	55.8	51.5
LCAF	**75.66**	**92.69**	**56.1**	**34.2**
LFPC [5]	74.46	92.04	60.8	-
HRank [20]	71.98	91.01	62.1	46.0
LCAF	**75.14**	**92.38**	**62.6**	**48.0**

5. Ablation Studies and Analysis

5.1. Effect of Weight Modification

LCAF with weight modification reduces the change in loss when pruning channels. To verify whether the weight modification is actually beneficial, we observed the difference in loss with and without weight modification by pruning each channel repeatedly. We independently removed each channel I_l^i $(i = 1, \cdots, 16)$ in a layer, and observed the difference in the validation loss of the network after the removal of each channel. The 9th convolutional layer of ResNet-32 on CIFAR-10 is used for this ablation, and the results are shown in Figure 3. For all channels, the loss change is significantly reduced. This proves the effectiveness of the weight modification. It is a useful technique that makes pruning more stable and results in faster convergence.

5.2. Independency on Number of Input Samples

Unlike the use of *filter* values, the use of *feature maps* necessarily depends on the input data distribution. Since we observe the linear approximation ability between the feature maps, the number of sample images used in the criterion calculation could behave as an important factor.

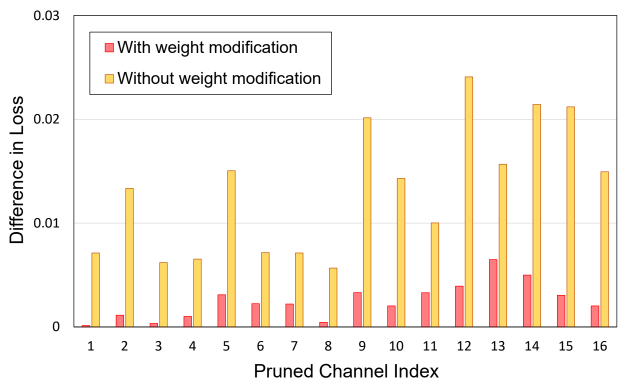

Figure 3. The ablation on weight modification. We observed the difference in validation loss by repeatedly removing each channel in a layer. With weight modification, the loss of the network varies much less.

Table 4. Ablation study on number of input samples used in LCAF calculation. The results show the accuracy after pruning with different numbers of input samples m.

CIFAR-10		CIFAR-100	
# samples	Acc (%)	# samples	Acc (%)
2	91.09	2	66.75
8	91.90	8	67.84
32	92.25	32	68.39
128	**92.44**	128	**68.52**
512	<u>92.38</u>	512	<u>68.41</u>
2048	92.34	2048	**68.52**

To verify the influence of the input data, we conducted an experiment using various numbers of input sample images. The same pruning process is applied while changing the number of images used in the LCAF criterion calculation to m = 2, 8, 32, 128, 512, and 2048. The performance after pruning is presented in Table 4. In this experiment, the ResNet-32 network and two datasets CIFAR-10 and CIFAR-100 are used, and all models are pruned with 55% FLOPs reduction.

In both CIFAR-10 and CIFAR-100, we obtain a considerably lower performance with 2 and 8 sample images. However, the accuracy converges after using 32 images. Although there are 50,000 training images, it appears that it is possible to extract the characteristics of each feature sufficiently by using only over 32 samples. This shows that our work is almost unaffected by the number of input samples. A more remarkable result is that a similar phenomenon is observed for both CIFAR-10 and CIFAR-100. Since CIFAR-100 has a considerably larger number of classes, fewer samples could not extract the appropriate features for CIFAR-100. However, these results prove that the criterion calculation of the proposed LCAF is effective regardless of the number of classes.

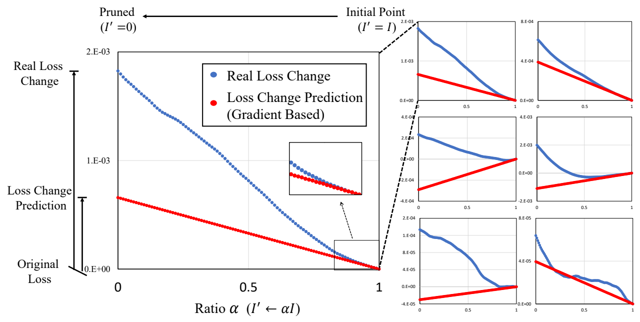

Figure 4. Observations of loss change by gradually decreasing the value of each feature to zero. (Right) Each graph shows the pruning of six different features. (Left) Magnified view of one graph. The horizontal line is the ratio α from 0 to 1, where we decrease the feature map value as $I' \leftarrow \alpha I$. The vertical line represents the loss change with respect to the original loss (at $I' = I$). The blue line indicates the actual loss change of the network when the feature gradually becomes zero. The red line represents the prediction from the gradient-based approach.

5.3. Global Pruning Criteria

In this section, we first reveal the problem of the gradient-based criterion by observing the actual loss change, and secondly, we show that the proposed normalized norm criterion is more effective quantitatively.

Observations on gradient-based approach. We conduct a simple experiment to observe the actual loss change and loss prediction based on Equation 13. The observations are shown in Figure 4. The six graphs on the right are the loss change graphs observed by gradually reducing each of the six different features to zero. That is, the feature map varies from the initial point ($I' = I$) to the pruned point ($I' = 0$). The figure on the left shows a magnified view of one graph. The red line indicates the loss change prediction based on the gradient at the initial point, whereas the blue line indicates the actual loss change recorded by forwarding the network at each point. As we can observe, the red line effectively approximates linearly at the initial point ($\alpha = 1$). However, it is not sufficient to predict the actual loss change when removing the channel.

Previously, this approach is known to be a more effective way than traditional norm-based criteria due to its clear theoretical background. However, the loss of the network is a complex function that includes many parameters. Therefore, the approximation using this gradient-based criterion is only suitable in an ideal situation, and has difficulties in providing accurate predictions in real settings.

Table 5. Ablation study on the pruning criteria. The networks are pruned based on three different global criteria.

	FLOPs reduction	
	50% ↓	60% ↓
Gradient-Based	93.32	92.97
Unnormalized Norm	92.36	90.02
Normalized Norm (Ours)	**93.66**	**93.45**

Quantitative Comparison. To validate the advantage of the proposed normalized norm criterion, three experiments were performed with the same training settings but different pruning criteria. 1) Gradient-based: Loss prediction using the Taylor series approximation expressed in Equation 13. 2) Unnormalized norm: Similar to our normalized norm criterion, but normalization is not used. Therefore, the measure of importance is $||\epsilon_l^i||$. 3) Normalized norm (proposed): The criterion used in the LCAF, which is expressed by Equation 11. The experiment was conducted with ResNet56 on CIFAR-10, and acceleration rates of 50% and 60% were used for diversity. We present the Top-1 accuracy of the three models in Table 5. Among the three criteria, our normalized norm criterion exhibits the best performance for both acceleration rates. This result validates the observation presented in Figure 4. As expected, the unnormalized norm (without normalization) yields considerably lower performance in both FLOPs reduction. Since the norm of the feature maps is affected by the depth of the layer, global pruning cannot proceed properly using the unnormalized norm criterion.

5.4. Computational Complexity

The calculation of the LCAF channel selection criterion seems complicated; however, it actually consists of simple matrix operations and does not require heavy computation, such as fine-tuning or back-propagation. Therefore, it does not take much time, although we observe the entire network.

We measured the actual computation time in our environment. For ResNet-50 on ImageNet, the criterion calculation for the entire network takes only 58 s to select the channel to prune. The following weight modification step takes even less than 0.1s since it only replaces the weight value. However, one epoch of fine-tuning takes 43 min, which is more dominant. In the entire process, the calculation for linear approximation requires an almost negligible amount of computation (2.27% of overall time).

5.5. Sub-network Architecture after pruning

Global pruning observes the entire network, unlike layer-wise pruning. Therefore, it has the advantage of finding optimal architecture. The final architecture of the compressed subnetwork is illustrated in Figure 5. The experiments are conducted with ResNet-32 on CIFAR-10, and 50% of the FLOPs are pruned. The black line indicates the original ResNet, and the red line indicates our final pruned model. We can observe that LCAF removed the channels similarly across the entire layer. One remarkable observation here is that there was not much pruning that occurred immediately after the downsampling layers (6 and 11). This indicates that there is much important information when downsampling occurs and the number of channels increases.

To verify the problem with the unnormalized norm criterion, we also observed the final pruned architecture when the unnormalized norm criterion is adopted (blue). As expected, pruning usually occurs in deeper layers, which have feature maps with small norms. Therefore, it finally yields an unstable structure that does not operate properly.

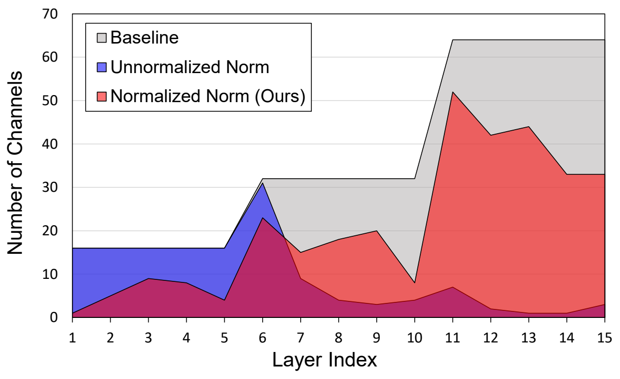

Figure 5. The final pruned architecture of proposed LCAF. Without normalization, pruning tends to remove deeper layers and does not prune the lower layer. The architecture becomes unstable and exhibits lower performance. There are 15 indices on the horizontal axis since ResNet-32 has only 15 residual blocks.

6. Conclusion

In this study, we propose a novel channel pruning framework, LCAF, which approximates each feature map by the linear combination of other feature maps. The LCAF effectively finds the replaceable channels by considering the correlation between the various features in a layer. In addition, weight modification was proposed to further reduce the loss when removing channels. Moreover, we propose a normalized norm criterion that overcomes the drawbacks of the previous gradient-based approach.

Acknowledgement

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

[1] Jonathan Frankle and Michael Carbin. The lottery ticket hypothesis: Finding sparse, trainable neural networks. *arXiv preprint arXiv:1803.03635*, 2018. 2

[2] Kai Han, Yunhe Wang, Qi Tian, Jianyuan Guo, Chunjing Xu, and Chang Xu. Ghostnet: More features from cheap operations. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1580–1589, 2020. 3

[3] Song Han, Jeff Pool, John Tran, and William Dally. Learning both weights and connections for efficient neural network. In *Advances in neural information processing systems*, pages 1135–1143, 2015. 1, 2

[4] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. 1, 5

[5] Yang He, Yuhang Ding, Ping Liu, Linchao Zhu, Hanwang Zhang, and Yi Yang. Learning filter pruning criteria for deep convolutional neural networks acceleration. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2009–2018, 2020. 5, 6

[6] Yang He, Guoliang Kang, Xuanyi Dong, Yanwei Fu, and Yi Yang. Soft filter pruning for accelerating deep convolutional neural networks. *arXiv preprint arXiv:1808.06866*, 2018. 5, 6

[7] Yihui He, Ji Lin, Zhijian Liu, Hanrui Wang, Li-Jia Li, and Song Han. Amc: Automl for model compression and acceleration on mobile devices. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 784–800, 2018. 2

[8] Yang He, Ping Liu, Ziwei Wang, Zhilan Hu, and Yi Yang. Filter pruning via geometric median for deep convolutional neural networks acceleration. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4340–4349, 2019. 1, 2, 5, 6

[9] Yihui He, Xiangyu Zhang, and Jian Sun. Channel pruning for accelerating very deep neural networks. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1389–1397, 2017. 2

[10] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015. 2

[11] Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017. 3

[12] 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. 2

[13] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4700–4708, 2017. 1

[14] Zehao Huang and Naiyan Wang. Data-driven sparse structure selection for deep neural networks. In *Proceedings of the European conference on computer vision (ECCV)*, pages 304–320, 2018. 6

[15] Donggyu Joo, Eojindl Yi, Sunghyun Baek, and Junmo Kim. Linearly replaceable filters for deep network channel pruning. In *The 34th AAAI Conference on Artificial Intelligence,(AAAI)*, 2021. 2

[16] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. Technical report, Citeseer, 2009. 2, 5

[17] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012. 1

[18] Bailin Li, Bowen Wu, Jiang Su, and Guangrun Wang. Eagleeye: Fast sub-net evaluation for efficient neural network pruning. In *European Conference on Computer Vision*, pages 639–654. Springer, 2020. 2

[19] Yawei Li, Shuhang Gu, Kai Zhang, Luc Van Gool, and Radu Timofte. Dhp: Differentiable meta pruning via hypernetworks. *arXiv preprint arXiv:2003.13683*, 2020. 2, 5

[20] Mingbao Lin, Rongrong Ji, Yan Wang, Yichen Zhang, Baochang Zhang, Yonghong Tian, and Ling Shao. Hrank: Filter pruning using high-rank feature map. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1529–1538, 2020. 1, 2, 5, 6

[21] Shaohui Lin, Rongrong Ji, Yuchao Li, Yongjian Wu, Feiyue Huang, and Baochang Zhang. Accelerating convolutional networks via global & dynamic filter pruning. In *IJCAI*, pages 2425–2432, 2018. 6

[22] 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 *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2790–2799, 2019. 5, 6

[23] Hanxiao Liu, Karen Simonyan, and Yiming Yang. Darts: Differentiable architecture search. *arXiv preprint arXiv:1806.09055*, 2018. 2

[24] Zhuang Liu, Jianguo Li, Zhiqiang Shen, Gao Huang, Shoumeng Yan, and Changshui Zhang. Learning efficient convolutional networks through network slimming. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2736–2744, 2017. 1

[25] Zechun Liu, Haoyuan Mu, Xiangyu Zhang, Zichao Guo, Xin Yang, Tim Kwang-Ting Cheng, and Jian Sun. Metapruning: Meta learning for automatic neural network channel pruning. *arXiv preprint arXiv:1903.10258*, 2019. 2

[26] Zhuang Liu, Mingjie Sun, Tinghui Zhou, Gao Huang, and Trevor Darrell. Rethinking the value of network pruning. *arXiv preprint arXiv:1810.05270*, 2018. 2

[27] Jian-Hao Luo, Jianxin Wu, and Weiyao Lin. Thinet: A filter level pruning method for deep neural network compression. In *Proceedings of the IEEE international conference on computer vision*, pages 5058–5066, 2017. 1, 2, 6

[28] Pavlo Molchanov, Stephen Tyree, Tero Karras, Timo Aila, and Jan Kautz. Pruning convolutional neural networks for resource efficient inference. *arXiv preprint arXiv:1611.06440*, 2016. 4

[29] 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. 2017. 5

[30] Hanyu Peng, Jiaxiang Wu, Shifeng Chen, and Junzhou Huang. Collaborative channel pruning for deep networks. In *International Conference on Machine Learning*, pages 5113–5122. PMLR, 2019. 5

[31] Hieu Pham, Melody Y Guan, Barret Zoph, Quoc V Le, and Jeff Dean. Efficient neural architecture search via parameter sharing. *arXiv preprint arXiv:1802.03268*, 2018. 2

[32] Adriana Romero, Nicolas Ballas, Samira Ebrahimi Kahou, Antoine Chassang, Carlo Gatta, and Yoshua Bengio. Fitnets: Hints for thin deep nets. *arXiv preprint arXiv:1412.6550*, 2014. 2

[33] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015. 2, 5

[34] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4510–4520, 2018. 6

[35] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014. 1

[36] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–9, 2015. 6

[37] Mingxing Tan and Quoc V Le. Efficientnet: Rethinking model scaling for convolutional neural networks. *arXiv preprint arXiv:1905.11946*, 2019. 3

[38] Junho Yim, Donggyu Joo, Jihoon Bae, and Junmo Kim. A gift from knowledge distillation: Fast optimization, network minimization and transfer learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4133–4141, 2017. 2

[39] Zhonghui You, Kun Yan, Jinmian Ye, Meng Ma, and Ping Wang. Gate decorator: Global filter pruning method for accelerating deep convolutional neural networks. *arXiv preprint arXiv:1909.08174*, 2019. 4

[40] Xiangyu Zhang, Xinyu Zhou, Mengxiao Lin, and Jian Sun. Shufflenet: An extremely efficient convolutional neural network for mobile devices. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6848–6856, 2018. 3

[41] Zhuangwei Zhuang, Mingkui Tan, Bohan Zhuang, Jing Liu, Yong Guo, Qingyao Wu, Junzhou Huang, and Jinhui Zhu. Discrimination-aware channel pruning for deep neural networks. In *Advances in Neural Information Processing Systems*, pages 875–886, 2018. 1, 2, 5, 6

[42] Barret Zoph and Quoc V Le. Neural architecture search with reinforcement learning. *arXiv preprint arXiv:1611.01578*, 2016. 2