# Batch Normalization Tells You Which Filter is Important

Junghun Oh Heewon Kim Sungyong Baik Cheeun Hong Kyoung Mu Lee
Department of ECE, ASRI, Seoul National University
{dh6dh, ghimhw, dsybaik, cheeun914, kyoungmu}@snu.ac.kr

## Abstract

*The goal of filter pruning is to search for unimportant filters to remove in order to make convolutional neural networks (CNNs) efficient without sacrificing the performance in the process. The challenge lies in finding information that can help determine how important or relevant each filter is with respect to the final output of neural networks. In this work, we share our observation that the batch normalization (BN) parameters of pre-trained CNNs can be used to estimate the feature distribution of activation outputs, without processing of training data. Upon observation, we propose a simple yet effective filter pruning method by evaluating the importance of each filter based on the BN parameters of pre-trained CNNs. The experimental results on CIFAR-10 and ImageNet demonstrate that the proposed method can achieve outstanding performance with and without fine-tuning in terms of the trade-off between the accuracy drop and the reduction in computational complexity and number of parameters of pruned networks.*

## 1. Introduction

Deep convolutional neural networks (CNNs) have shown remarkable performance in various computer vision tasks [26, 32, 6], however CNNs often require large memory storage and expensive computation costs. Such high demand for memory and computation limits their applications in resource-constrained environment, such as mobile devices. Among various approaches that tackle this problem, filter pruning approaches have gained attention in recent years as one of prospective approaches for effectively reducing memory storage and computation costs on general tensor processors.

Filter pruning aims to find and remove unimportant filters that do not contribute much to the final output of CNNs. Owing to the simplicity, filter-weight-based pruning [20, 12] is one of the most widely used approaches in that other model compression methods use it for aiding decision processes on which filters to prune [38, 15, 9, 8, 19, 35]. The simplicity is achieved by using readily available information (*e.g.* $\ell_1$-norm of the filter weights) that can be easily computed from off-the-shelf pre-trained networks without processing of training data or pruning-tailored training. However, relying only on the statistics of filters that have relatively indirect information about CNN outputs, compared to gradients or output feature maps, can render filter-weight-based pruning rather limited [37, 28], resulting in a significant degradation in performance after pruning.

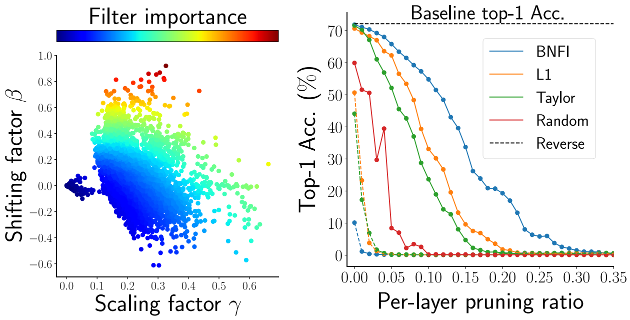


Figure 1: Our proposed filter importance measure is illustrated in the left figure. The figure visualizes how pre-trained batch normalization (BN) parameters, $\gamma$ and $\beta$, are mapped into the filter importance. The figure on the right shows how the performance of pre-trained MobileNetV2 changes as its filters are removed based on each filter importance measure. For each per-layer pruning ratio, the performance degradation happens the least when pre-trained CNNs are pruned with our proposed filter importance measure (BNFI), indicating that an accurate filter importance can be determined by pre-trained BN parameters.

Several works have shifted the attention to other information that is more descriptive with respect to the final output of CNNs, such as activation outputs [14, 13, 27, 22] or gradients of filter weights [28, 29]. Although using such more informative statistics has led to higher final performance after pruning, they have sacrificed the simplicity in the process. In particular, they require a number of network feed-forward processes using training data to obtain activation outputs or gradient values. As such, these data-dependent methods are not suitable as off-the-shelf filter pruning criteria that other advanced pruning algorithms [38, 15, 9, 8, 19, 35] could depend on.

In this work, we claim that rich information regarding the filter importance can be obtained directly from pre-trained networks without processing of training data or pruning-tailored training. In particular, we show that the distribution of activation outputs can be estimated with BN parameters of pre-trained networks. Such estimation allows us to propose an accurate yet simple-to-compute filter pruning criterion, dubbed as BNFI (**B**atch **N**ormalization tells you which **F**ilter is **I**mportant), using the pre-trained BN parameters, as illustrated in Figure 1.

To validate the effectiveness of the proposed method, we conduct extensive experiments on CIFAR-10 and ImageNet dataset with various network architectures. We show that the proposed criterion can sort the filters in order of importance more accurately than the filter-weight-based methods and even the data-dependent methods, especially on MobileNetV2 [33]. Combined with a simple greedy per-layer pruning ratio search strategy using the proposed filter importance, the proposed method also achieves better or competitive results after fine-tuning, compared to the existing complex learning-based pruning methods.

We summarize the contributions of this paper as follows:

- Without data-dependent computation, the proposed method can estimate the feature distribution of activation outputs using the BN parameters, enabling the measurement of filter importance in terms of the BN parameters.

- Experimental results without fine-tuning stages demonstrate that the proposed method outperforms the existing filter pruning methods especially on MobileNetV2.

- Using the proposed filter importance, we propose a simple greedy per-layer pruning ratio search strategy and show comparable or better performance after fine-tuning compared to the complex learning-based methods.

## 2. Related work

Filter pruning aims to eliminate unnecessary filters in CNNs. The filter pruning methods can be roughly categorized into two groups based on whether they perform pruning-tailored training or importance estimation. The former aims to induce redundant filters via a specialized learning framework while the latter aims to estimate the importance of each filter with respect to the performance of pre-trained networks. This work can be considered as one of the importance estimation methods, which we further classify them into three groups: filter-weight-based methods, activation-based methods, and gradient-based methods.

The filter-weight-based methods [20, 12] use the filter weight values to estimate the filter importance. Li *et al.* [20] determine the filter importance via $\ell_1$-norm of the filter weights. He *et al.* [12] propose to prune the filters near the geometric median of the filters in each layer. These filter-weight-based methods are widely applicable since the filter importance can be easily evaluated from a pre-trained network without any data-dependent computation. Owing to its easy-access property, many works adopt the filter-weight-based criterion onto their model compression frameworks [38, 15, 9, 8, 19, 35].

The major drawback of the filter-weight-based methods is that they have resorted to simple but indirect measures and neglected other operations in CNNs, such as batch normalization (BN) and non-linear activation function, that can impact the final output of CNNs more directly. To overcome this problem, the activation-based methods [14, 27, 13, 22, 4] and the gradient-based methods [3, 28] focus on the activation outputs and gradient values of filters. He *et al.* [13] and Luo *et al.* [27] find redundant activation channels via complex optimization methods that involve training data. Dubey *et al.* [4] prune the filters with the low $\ell_1$-norm of corresponding activation values that are computed over training data before their proposed corset-based compression stage. Molchanov *et al.* [28] compute the gradient value of each weight through several network feed-forward processes and estimate the importance of filters via Taylor expansions. Despite their notable pruning performance, these data-dependent methods suffer from the heavy computation required to find unimportant filters, limiting their applicability.

In this work, we claim that the accurate filter importance can be measured without relying on data for the heavy computation of activation values or optimization processes. Specifically, we show that the activation outputs (output feature maps of non-linear activation functions) can be estimated using BN parameters of pre-trained networks.

Our work may be considered to be similar to sparsity learning methods that use BN layers to guide filter pruning [23, 37, 39, 17]. However, they still require training to use BN parameters to intentionally induce redundant activation channels. Furthermore, these methods do not consider the impact of non-linear activation functions into their proposed filter importance [23, 37, 39] or only focus on ReLU activation function [17]. By contrast, we show that an accurate and flexible filter importance measure can be achieved by using BN parameters of pre-trained networks. Specifically, our method is applicable to general activation functions and considers the impact of activation functions for more accurate estimation of the filter importance.

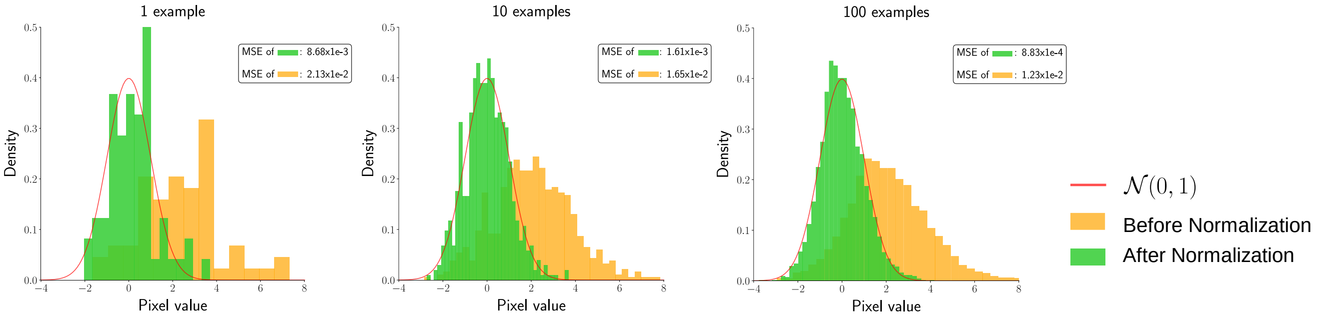


Figure 2: Plots of probability density function of standard normal distribution and normalized histogram of pixel values before normalization and after normalization for a different number of input images. These results are obtained from a channel in a certain layer in ResNet-56 on 1 (left), 10 (middle), and 100 (right) CIFAR-10 test images. The MSE(Mean Squared Error) score is measured between the histogram and the Gaussian probability distribution function. The average MSE score after normalization for all channels in the same layer is $4.2 \times 10^{-4}$ on 100 input images.

## 3. Method

Motivated by our observation that the batch normalization (BN) parameters of pre-trained networks contain meaningful information about the activation outputs, we propose to measure the importance of filters in terms of pre-trained BN parameters without relying on data. In the following subsections, we define the filter pruning problem in Section 3.1, estimate the distribution of activation outputs and define the filter importance using the BN parameters in Section 3.2, and propose a simple greedy method to find per-layer pruning ratios using the proposed filter importance.

### 3.1. Background

**Batch normalization.** Batch normalization (BN) [16] layer is a commonly used module to facilitate the training process of deep neural networks. BN is often placed after the convolution operation whose output feature map is denoted as $\boldsymbol{x} \in \mathbb{R}^{B \times C \times H \times W}$, where $B$, $C$, $H$, and $W$ denote the size of mini-batch, the number of channels, and height and width of the feature map, respectively. Batch normalization performs normalization and affine transformation for each $i$-th channel $\boldsymbol{x}_i$:

$$\hat{\boldsymbol{x}}_i = \frac{\boldsymbol{x}_i - \mu_B}{\sqrt{\sigma_B^2 + \epsilon}}, \tag{1}$$

$$\boldsymbol{x}_{BN,i} = \gamma_i \cdot \hat{\boldsymbol{x}}_i + \beta_i, \tag{2}$$

where $\mu_B$ and $\sigma_B$ respectively denote the mean and standard deviation values of $\boldsymbol{x}_i$ over a mini-batch of size $B$; $\epsilon$ is an arbitrary small constant for numerical stability; and $\gamma_i$ and $\beta_i$ are learnable parameters for each $i$-th channel, respectively.

**Activation function.** In commonly used modern CNNs, such as VGGNet [34] and ResNet [6], a BN layer is followed by a non-linear activation layer. Let $g(\cdot)$ denote the non-linear activation function. Then, the final activation output $\boldsymbol{z}$, which has the same dimension as $\boldsymbol{x}$, is formally given by,

$$\boldsymbol{z}_i = g(\boldsymbol{x}_{BN,i}). \tag{3}$$

Note that the activation output $g(\boldsymbol{x}_{BN,i})$ is fed into the next convolutional layer as input.

**Problem definition.** Let $\boldsymbol{\mathcal{W}}$ denote a set of filters of a convolutional layer. When pruning $k$ filters in the layer, the optimal set of $k$ unimportant filters is defined as follows:

$$\boldsymbol{\mathcal{W}}_k^s \equiv \underset{\boldsymbol{\mathcal{W}}_k}{\operatorname{argmin}}\, \mathcal{L}(\boldsymbol{D}, \boldsymbol{\mathcal{W}} - \boldsymbol{\mathcal{W}}_k), \tag{4}$$

where $\boldsymbol{D}$, $\mathcal{L}$ and $\boldsymbol{\mathcal{W}}_k$ denote the target dataset, the target loss function, and a set of filters with the size of $k$, respectively. (In Equation (4), the filters of the other layers remain intact.) Unfortunately, finding $\boldsymbol{\mathcal{W}}_k^s$ requires going through measuring the performance of all possible networks after pruning $k$ filters of the given layer ($\binom{n}{k}$ network inferences), which are intractable computations particularly in recent heavy networks or dataset. Therefore, many methods have attempted to approximately sort the filters in the order of the estimated relative importance of each filter to obtain the approximate set of $k$ unimportant filters:

$$\boldsymbol{\mathcal{W}}_k^s \approx \{\mathcal{W} \mid \mathcal{I}_{\mathcal{W}} \le \mathcal{I}^k\}, \tag{5}$$

where $\mathcal{I}_{\mathcal{W}}$ and $\mathcal{I}^k$ denote the importance of filter $\mathcal{W}$ and $k$-th smallest importance, respectively. Among them, the activation-based methods focus on the activation outputs to determine the filter importance:

$$\mathcal{I}_{\mathcal{W}_i} \equiv \mathcal{I}(\boldsymbol{z}_i), \tag{6}$$

where $\mathcal{I}(\cdot)$ denotes an importance evaluation function.

### 3.2. Filter importance from BN parameters

**Gaussian assumption.** We start with the assumption on the output distribution of a BN layer. By Equation (1), the input of a BN layer $\boldsymbol{x}_i$ is normalized to $\hat{\boldsymbol{x}}_i$ each element of which, $\hat{x}_i$, has a mean of 0 and a standard deviation of 1. Under the assumption that the batch size $B$ is sufficiently large, we can approximate the distribution of $\hat{x}_i$ as the Gaussian distribution according to the Central Limit theorem, $\hat{x}_i \sim \mathcal{N}(0, 1)$. Figure 2 demonstrates that our assumption holds well with a larger batch size.

**Estimating the distribution of activation outputs.** Under the Gaussian assumption, we can formulate the outputs of subsequent operations in terms of $\gamma_i$ and $\beta_i$. The following affine transformation in Equation (2) shifts and scales the mean and the standard deviation by a factor of $\beta_i$ and $|\gamma_i|$, respectively, leading to $x_{BN,i} \sim \mathcal{N}(\beta_i, \gamma_i^2)$. Since we can estimate the distribution of $x_{BN,i}$ in terms of $\gamma_i$ and $\beta_i$, we also can estimate that of the activation outputs using BN parameters. For example, if the activation function is ReLU, $g(x) = max(x, 0)$, $z_i$ can be estimated by the truncated Gaussian distribution. Finally, $x_{BN,i}$ is transformed by a non-linear activation function $g$, producing $z_i = g(x_{BN,i})$. This suggests that the distribution of activation outputs can be estimated by using the BN parameters since $g$ is known in a target pre-trained network.

**Definition of filter importance.** With the Gaussian assumption and the subsequent formulation of activation output distributions in terms of BN parameters, we propose a novel definition of filter importance using the estimated activation output distribution. For our definition of filter importance, we assume that an activation output channel with large absolute values is important because it can have significant impact on the output of the subsequent layers and thus the final output [27, 4, 13]. From this assumption, we define the filter importance as *the expectation of absolute values*:

$$\mathcal{I}(\boldsymbol{z}_i) \equiv \int_{-\infty}^{\infty} |g(z)| \cdot f(z; \beta_i, |\gamma_i|)\, dz, \tag{7}$$

where $f(\cdot; \beta, \gamma)$ denotes a Gaussian probability density function with a mean of $\beta$ and a standard deviation of $\gamma$. For computational feasibility, the infinite integration bound can be replaced with small values (e.g. $\pm 5$) with a negligible error and the integration can be easily computed with libraries, such as *SciPy*.

**Filter importance on sparse activation** Although the proposed filter importance in Equation (7) can be applied to any activation function, we empirically found that it is better to consider the sparsity (ratio of zeros) of activation outputs when there is high sparsity on activation outputs (e.g. when the activation function is ReLU). When the sparsity is high, meaningful information that could be present in non-zero values is dominated by the large number of zero values according to the filter importance defined in Equation (7). In other words, in the case of spare activation outputs, the corresponding filter should be considered to be important if the expectation of non-zero values is large enough, while it is considered to be unimportant according to Equation (7) due to the dominant number of zero values. Therefore, we slightly modify the definition of filter importance in Equation (7) to reduce the impact of the zero values in the case of high sparsity:

$$\begin{aligned} \mathcal{I}(\boldsymbol{z}_i) &\equiv \int_{-\infty}^{\infty} |g(z)| \cdot \frac{f(z; \beta_i, |\gamma_i|)}{\mathcal{N}}\, dz, \\ where\ \ \mathcal{N} &= \int_{g(z)\neq 0} f(z; \beta_i, |\gamma_i|)\, dz, \end{aligned} \tag{8}$$

Equation (8) can be seen as a conditional expectation on positive regions. Note that Equation (8) becomes Equation (7) when the activation function does not induce sparsity. Experiments on the impact of sparsity are provided in the supplementary material.

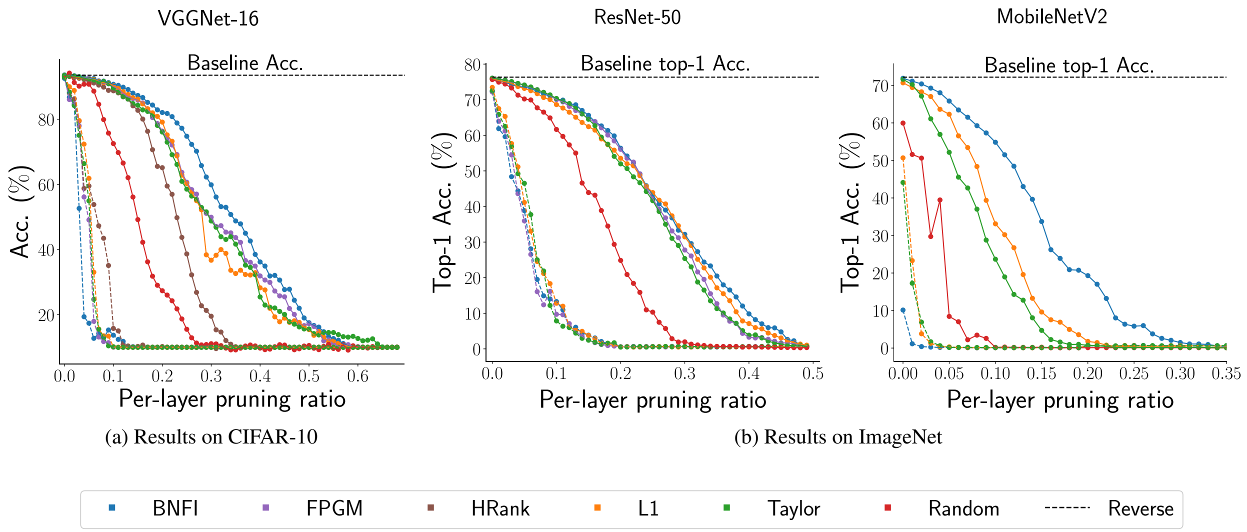
### 3.3. Per-layer pruning ratio search

The performance of a pruned network is largely affected by the structure of the resulting network architecture (e.g. the number of remaining filters in each layer) [25, 19, 24]. Therefore, it is important to carefully design decision processes on how many filters to prune for each layer. However, filters are interdependent in that removing any filter of a layer can affect other layers. As it is computationally infeasible to find an optimal number of filters to prune for each layer due to the interdependence and the large number of filters, we propose a greedy method to determine how many filters to prune in each layer based on the proposed filter importance measure.

In particular, we determine the number of filters to prune for each layer based on the contribution of each layer to the final performance of a pre-trained network. The contribution of the $i$-th layer is measured as the performance degradation $\delta_i$ after pruning a certain portion (pruning ratio $r_i$) of filters in the layer (where filters with the least importance are pruned first according to Equation (8)), where $\delta_i$ can be calculated as the difference between the validation accuracy of a pre-trained network $V_{base}$ and that of a resulting pruned network $V_i'$:

$$\delta_i = |V_{base} - V_i'(r_i)|. \tag{9}$$

Based on the relation between the performance degradation and pruning ratio outlined in Equation (9), we find a pruning ratio $r_i$ for each $i$-th layer that gives a pruned network whose performance degradation closely matches $\delta_i$, which is given as a hyper-parameter. To find $r_i$ efficiently, we conduct an iterative algorithm similar to binary search. Given a lower bound of $l$ and an upper bound of $u$, we measure the validation accuracy $V_i'(\frac{l+u}{2})$, compare the degraded performance to $\delta_i$, and modify $l$ or $u$ according to the comparison result. After 5 iteration, we select $\frac{l+u}{2}$ as a resulting pruning ratio.



(a) Results on CIFAR-10

(b) Results on ImageNet

Figure 3: Experimental results on CIFAR-10 and ImageNet without fine-tuning. For each network, we estimate the importance of filters using corresponding method and sort the filters in order of the importance. Then, according to the pruning ratio, we prune the filters in ascending order of importance. For each method, we also present the results on reverse pruning where the filters are removed in descending order of importance. For the Random method, the filter importance is randomly determined and the results are averaged over 3 experiments. All layers of a pruned network have the same pruning ratio, which is represented in the x-axis.

# 4. Experiment

## 4.1. Experimental details

**Datasets.** We use CIFAR-10 [18] and ImageNet datasets [2] to evaluate the classification performance of the baseline and pruned models. CIFAR-10 are widely used dataset that contains 50,000 training and 10,000 test images with 10/100 classes and 32 $\times$ 32 resolution. ImageNet is a large-scale and challenging dataset which consists of 1,281,167 training and 50,000 validation images with 1,000 classes. For data augmentation, we conduct random cropping and flipping on all training sets and center cropping for the validation set of ImageNet.

**Models.** We use the recent commonly used CNNs models, such as VGGNet [34], ResNet [6, 7], and MobileNetV2 [33]. For the experiments on CIFAR-10 dataset, we use the light version of VGGNet-16, as in [20] and [23]. For the experiments on ImageNet dataset, we use ResNet-50 and MobileNetV2 to validate the effectiveness of the proposed method on such complex or light models.

**Training Settings.** For VGGNet-16 and ResNet-50, we train the baseline models with the batch size of 64 for 160 epochs on CIFAR-10 and the batch size of 256 for 120 epochs on ImageNet datasets. In the fine-tuning stage, the settings are the same except that fewer epochs are used (120 for CIFAR-10 and 80 for ImageNet datasets). We use the stochastic gradient descent algorithm for optimizer with weight decay $10^{-4}$ and momentum 0.9, and the learning rate is initially set to 0.1 and decayed by 0.1 at one-half and one-quarter of total epochs. In the case of MobileNetV2, we use weight decay $5 \times 10^{-4}$ and momentum 0.9, and the learning rate is scheduled by cosine annealing scheduler, updated from 0.05 to 0.

**Evaluation Method.** For the evaluation of pruning performance, we use three metrics: the accuracy drop with respect to the accuracy of baseline models, the ratio of the pruned model parameters, and computational complexity. Instead of the final accuracy of the pruned model, we use the accuracy drop as a performance measure of the pruned model since the results of previous methods have slightly different baseline accuracy. We use floating point operations (FLOPs) as the measurement of computational complexity. As for the ratio of the pruned parameters, we count the number of parameters and FLOPs only for the convolutional, following the settings from [12, 10]Using the three metrics, we mainly compare the proposed pruning method to the importance estimation method [20, 12, 22, 28] without fine-tuning settings to demonstrate that the proposed filter importance BNFI better identifies the network redun-

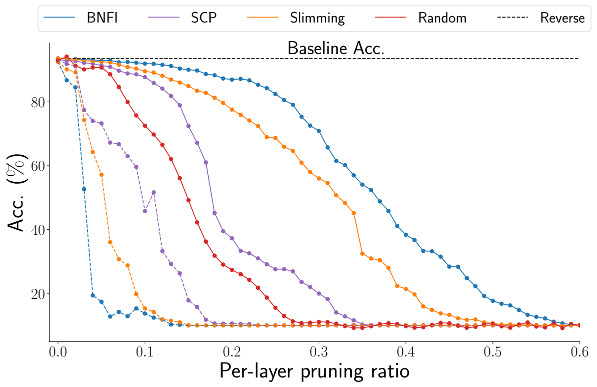
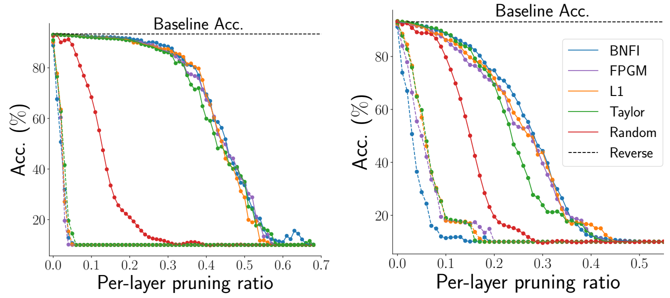


Figure 4: Comparison with the BN-based methods without fine-tuning. All details are same with the experiments on Figure 3. The baseline network and the dataset is VGGNet-16 and CIFAR-10.



(a) Swish (b) Leaky ReLU

Figure 5: Experimental results on other activation functions. All details are same with the experiments on Figure 3. The baseline networks are VGGNet-16 with a different activation function.

dancy.

**Implementation details.** For both ResNet-50 and MobileNetV2, where each residual block has 1x1-3x3-1x1 convolutional layers, we only prune the 3x3 convolutional layer since the computations of 1x1 convolutional layers are relatively light. To reproduce the results of the gradient-based method [28], we used the entire training data to accumulate the gradient values. We used the available source code to reproduce the activation-based method [22]. When applying the proposed per-layer pruning ratio strategy, we use the training accuracy instead of the validation accuracy since validation data is not available. In the case of CIFAR-10 dataset, we obtain the final pruned network through several searching-pruning-fine-tuning processes to carefully prune the network. We use the entire training data for CIFAR-10 and 50,000 selected training data for ImageNet for the strategy. All experiments are conducted on RTX 2080Ti GPUs.

## 4.2. Pruning without fine-tuning

In this subsection, we present the pruning results without fine-tuning stage to demonstrate the effectiveness of the proposed filter importance for determining redundant filters in pre-trained networks. We compare BNFI with the existing filter-weight-based methods (L1 and FPGM) [20, 12], the activation-based method (HRank) [22], and the gradient-based method (Taylor) [28]. Note that HRank and Taylor are data-dependent methods. For simplicity, let DOI and AOI denote "pruning in a descending order of importance" and "pruning in an ascending order of importance", respectively.

**Results on CIFAR-10.** Figure 3a shows the results on CIFAR-10. Across most of the pruning ratios, the performance degradation of BNFI is less in AOI and more in DOI by a significant margin. These results mean that BNFI can find both important filters and unimportant filters better than the other methods. We want to note that even compared to HRank and Taylor, which are data-dependent methods, BNFI achieves outstanding results. These results demonstrate that the proposed method can find the unimportant filters more accurately even without data-driven information.

**Results on ImageNet.** We conduct experiments on much more complex dataset, ImageNet dataset. The results are presented in Figure 3b. For ResNet-50, BNFI shows almost same or slightly less accuracy drop until the pruning ratio of 0.3 but outperforms the existing methods in pruning ratio larger than 0.3 in AOI, which means the proposed method also works well on such a complex model. Remarkably, BNFI significantly outperforms L1 and Taylor in both AOI and DOI in MobileNetV2. In AOI, the accuracy of BNFI in pruning ratio from 0.1 to 0.15 is higher than 30% points compared to the results of the other methods. Further, when the pruning ratio is merely 0.01, the accuracy is destructively degraded by BNFI in DOI, 72.19% to almost 10% and almost 0% after pruning ratio of 0.02. These outstanding results demonstrate that BNFI can accurately identify important and unimportant filters in depth-wise convolutional layers much better than L1 and Taylor. These experimental results suggest that BNFI can be used to investigate the redundancy in such a recent efficient architecture.

**Comparision with the BN-based methods.** Similar to our work, the existing BN-based methods [23, 37, 39, 17] also determine the importance of filter via BN parameters. However, our proposed filter importance measure BNFI is a

Table 1: Experimental results on the impact of pruning criterion onto the final performance after fine-tuning. The pre-trained MobileNetV2 is pruned by each criterion and fine-tuned on ImageNet dataset. The pruning ratio is shared across the entire layers. For the Random criterion, we report the mean and standard deviation of 3 experiments. The Reverse method determines the filter importance using the proposed method and prune the filters in order of importance.

| Pruning ratio | Criterion | Acc.(%) |
|---|---|---|
| 0.25 | BNFI | **71.59** |
| | L1 | 71.50 |
| | Random | 71.37(±0.05) |
| | Reverse | 70.64 |
| 0.5 | BNFI | **69.65** |
| | L1 | 69.53 |
| | Random | 69.35(±0.06) |
| | Reverse | 68.30 |

more accurate criterion than other existing BN-based methods, which require an ad-hoc learning process to forcibly induce unimportant filters. To support our claim, we compare with other BN-based methods in terms of the accuracy without fine-tuning, as illustrated in Figure 4. In AOI, BNFI causes a minor accuracy drop, less than 10%, until the pruning ratio of 0.25. On the other hand, Slimming and SCP significantly degrade the network performance as the pruning ratio increases, particularly in the case of SCP. Notably, in the pruning ratio of 0.2, BNFI still maintains the original performance, about 90%, but Slimming and SCP achieve 80% and 40%, respectively. In DOI, BNFI causes a destructive damage on the performance, about 20% in the pruning ratio of 0.05, but Slimming and SCP still show a good accuracy, about 75%. The overall results demonstrate that BNFI is more accurate than Slimming and SCP, meaning that the proposed method utilizes the concealed information much better than the existing BN-based methods.

**Generalization to other activation functions.** In this section, we show the proposed filter importance is also effective for other activation functions, such as Swish [31] and Leaky ReLU [36]. To validate the generality, we conduct experiments without fine-tuning on pre-trained networks with different activation functions, which is shown in Figure 5. In the case of Swish, BNFI shows comparable or slightly good performance on average in AOI and almost no accuracy drop until the pruning ratio of 0.2. These results demonstrate that BNFI can estimate the distribution of activation outputs, and thus the importance quite accurately even when the activation function is Swish. BNFI shows quite better results on Leaky ReLU. BNFI causes more steep drop of accuracy than the other methods in all of the pruning ratios in DOI. In AOI, BNFI shows rather good results in the pruning ratio from 0.2 to 0.3, more than 5% points than the other methods, demonstrating BNFI also can be applied to Leaky ReLU. All of these results show that the proposed method is applicable to other activation functions.

### 4.3. Pruning with fine-tuning

Although the focus of this paper is to develop a novel filter importance measure, we compare our method to more complex methods with pruning-tailored learning process to show the potential of BNFI. We sort the filters of a pre-trained network according to BNFI and search per-layer pruning ratios using the method in Section 3.3. Then, we prune the full network and fine-tune for several epochs.

**Impact on after fine-tuning results.** Before the discussion on the main results, we want to discuss the correlation between the performance without and with fine-tuning to demonstrate that the final performance is highly affected by which filters to prune, demonstrating the value of our work. The experimental results are shown in Table 1. The only difference of each result with same pruning ratio is which filters are removed. As shown in Figure 3, the performance without fine-tuning is much better when the filter importance is measured by BNFI. Similar to the results without fine-tuning, the results after fine-tuning significantly differ in the final accuracy, about 1% points between the results of BNFI and Reverse. These results indicate that the accurate filter importance is valuable for achieving higher performance even after fine-tuning.

**Results on CIFAR-10.** Table 2 shows the experimental results after fine-tuning on CIFAR-10 dataset. Through several searching-pruning-fine-tuning processes, the proposed method found a very efficient architecture whose number of parameters and FLOPs are pruned by 97.07% and 74.81%, respectively, which is most efficient result among the presented results. Despite such a huge reduction ratio, the accuracy drop is merely 0.04%. These results are outstanding compared to the sparsity learning methods (Slimming, Variational Pruning and SCP) and to the importance estimation methods (L1 and HRank), demonstrating the effectiveness of the proposed filter importance for determining where to prune and how many to prune.

**Results on ImageNet.** To further evaluate BNFI on a larger dataset, we also conduct experiments on ImageNet, which is presented in Table 3. For ResNet-50, we provide three results with different FLOPs. BNFI shows an accuracy drop of 0.16 when FLOPs are 3.02G, which is better than the sub-network search method, MetaPruning. In the intermediate pruning ratio, BNFI acheives an accuracy drop of 0.86, which is outstanding results compared to the other methods. For the most efficient results, BNFI also

Table 2: Pruning and fine-tuning results on CIFAR-10. The notation ”↓” means the reduction rate of the corresponding metrics.

| Model | Method | Baseline Acc. (%) | Acc. (%) | Acc. Drop (%) | Parameters ↓ (%) | FLOPs ↓ (%) |
|---|---|---|---|---|---|---|
| VGGNet-16 | Slimming [23]* | 93.85 | 92.91 | 0.94 | 87.97 | 48.12 |
| | Variational Pruning [39] | 93.25 | 93.18 | 0.07 | 73.34 | 39.10 |
| | SCP [17] | 93.85 | 93.79 | 0.06 | 93.05 | 66.23 |
| | L1 [20] | 93.25 | 93.40 | −0.15 | 64.00 | 34.20 |
| | HRank [22] | 93.96 | 93.43 | 0.53 | 82.90 | 53.50 |
| | **BNFI (Ours)** | 93.50 | 93.46 | 0.04 | 94.07 | 74.81 |

[*] Reproduced results in [17].

Table 3: Pruning and fine-tuning results on ImageNet.

| Model | Method | Baseline Top-1 Acc. (%) | Top-1 Acc. (%) | Top-1 Acc. Drop (%) | FLOPs |
|---|---|---|---|---|---|
| ResNet-50 | MetaPruning [24] | 76.60 | 76.20 | 0.40 | 3.00G |
| | **BNFI (Ours)** | 76.33 | 76.17 | 0.16 | 3.02G |
| | SFP [10] | 76.15 | 74.61 | 1.54 | 2.39G |
| | ThinNet [27] | 72.88 | 72.04 | 0.84 | 2.90G |
| | HRank [22] | 76.15 | 74.98 | 1.17 | 2.30G |
| | Taylor [28] | 76.18 | 74.50 | 1.68 | 2.25G |
| | DSA [30] | 76.02 | 74.10 | 0.92 | 2.47G |
| | **BNFI (Ours)** | 76.33 | 75.47 | 0.86 | 2.34G |
| | MetaPruning [24] | 76.60 | 75.40 | 1.20 | 2.01G |
| | HRank [22] | 76.15 | 71.98 | 4.17 | 1.55G |
| | FPGM [12] | 76.15 | 74.83 | 1.32 | 1.92G |
| | DCP [40] | 76.01 | 74.95 | 1.06 | 1.83G |
| | Hinge [21] | 76.10 | 74.70 | 1.40 | 1.88G |
| | DSA [30] | 76.02 | 74.69 | 1.33 | 2.06G |
| | **BNFI (Ours)** | 76.33 | 75.02 | 1.29 | 1.94G |
| MobileNetV2 | MetaPruning [24] | 72.00 | 71.20 | 0.80 | 217M |
| | AMC [11] | 71.80 | 70.80 | 1.00 | 220M |
| | LeGR [1] | – | 71.40 | – | 224M |
| | **BNFI (Ours)** | 72.19 | 70.97 | 1.22 | 220M |
| | DMC [5] | 71.88 | 68.37 | 3.51 | 162M |
| | LeGR [1] | – | 69.40 | – | 160M |
| | **BNFI (Ours)** | 72.19 | 68.72 | 3.47 | 158M |

shows considerably great results than the complex learning-based methods, such as DSA and Hinge. The above results demonstrate that the proposed filter importance is effective in determining where to prune and how many to prune, even on such a large dataset and a complex network. We also present the results on MobileNetV2. In MobileNetV2, BNFI achieves comparable or better results to the learning-based methods, an accuracy drop of 1.22 with FLOPs of 220M and an accuracy drop of 3.47 with FLOPs of 158M, which is slightly better than the result of DMC. Considering that the results of the learning-based methods are due to the learning process to learn where and how to prune simultaneously, the results show the potential of BNFI since the two major factors are only determined by BNFI.

## 5. Conclusion

In this paper, we propose a novel filter pruning criterion, BNFI, by extracting the concealed information in the BN parameters. From the Gaussian assumption, we estimate the distribution of activation outputs, leading to our definition of the filter importance in terms of the BN parameters. Since BNFI considers the impact of the BN layer and the activation function, it enables more accurate filter pruning decision than the existing filter-weight-based methods, while maintaining the easy-access property. The experimental results without fine-tuning on various models and datasets demonstrate that BNFI can find unimportant and important filters more accurately than the existing methods, especially on MobileNetV2. Furthermore, we show that the proposed filter importance can be used to search the per-layer pruning ratio, exhibiting the comparable or better results after fine-tuning compared to the complex learning-based methods.

**Acknowledgment** This work was supported in part by IITP grant funded by the Korean government (MSIT) [NO.2021-0-01343, Artificial Intelligence Graduate School Program (Seoul National University)]

## References

[1] Ting-Wu Chin, Ruizhou Ding, Cha Zhang, and Diana Marculescu. Towards efficient model compression via learned global ranking. In *CVPR*, 2020.

[2] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and d Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *CVPR*, 2009.

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

[4] Abhimanyu Dubey, Moitreya Chatterjee, and Narendra Ahuja. Coreset-based neural network compression. In *ECCV*, 2018.

[5] Shangqian Gao, Feihu Huang, Jian Pei, and Heng Huang. Discrete model compression with resource constraint for deep neural networks. In *CVPR*, 2020.

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

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

[8] Yang He, Yuhang Ding, Ping Liu, Linchao Zhu, Hanwang Zhang, and Yi Yang. Learning filter pruning criteria for deep convolutional neural networks acceleration. In *CVPR*, 2020.

[9] Yang He, Xuanyi Dong, Guoliang Kang, Yanwei Fu, Chenggang Yan, and Yi Yang. Asymptotic soft filter pruning for deep convolutional neural networks. *IEEE Transactions on Cybernetics*, pages 3594–3604, 2019.

[10] Yang He, Guoliang Kang, Xuanyi Dong, Yanwei Fu, and Yi Yang. Soft filter pruning for accelerating deep convolutional neural networks. In *IJCAI*, 2018.

[11] 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 *ECCV*, 2018.

[12] Yang He, Ping Liu, Ziwei Wang, Zhilan Hu, and Yi Yang. Filter pruning via geometric median for deep convolutional neural networks acceleration. In *CVPR*, 2019.

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

[14] Hengyuan Hu, Rui Peng, Yu-Wing Tai, and Chi-Keung Tang. Network trimming: A data-driven neuron pruning approach towards efficient deep architectures. In *ICLR*, 2016.

[15] Gao Huang, Shichen Liu, Laurens van der Maaten, and Kilian Q. Weinberger. Condensenet: an efficient densenet using learned group convolutions. *arXiv preprint arXiv:1711.09224*, 2017.

[16] Sergey Ioffe and Christian Szegedy. Batch normalization: accelerating deep network training by reducing internal covariate shift. In *ICML*, 2015.

[17] Minsoo Kang and Bohyung Han. Operation-aware soft channel pruning using differentiable masks. In *ICML*, 2020.

[18] Alex Krizhevsky. Learning multiple layers of features from tiny images. *University of Toronto*, 05 2012.

[19] Bailin Li, Bowen Wu, Jiang Su, and Guangrun Wang. Eagleeye: fast sub-net evaluation for efficient neural network pruning. In *ECCV*, 2020.

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

[21] Yawei Li, Shuhang Gu, Christoph Mayer, Luc Van Gool, and Radu Timofte. Group sparsity: the hinge between filter pruning and decomposition for network compression. In *CVPR*, 2020.

[22] Mingbao Lin, Rongrong Ji, Yan Wang, Yichen Zhang, Baochang Zhang, Yonghong Tian, and Ling Shao. Hrank: filter pruning using high-rank feature map. In *CVPR*, 2020.

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

[24] 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. In *ICCV*, 2019.

[25] Zhuang Liu, Mingjie Sun, Tinghui Zhou, Gao Huang, and Trevor Darrell. Rethinking the value of network pruning. In *ICLR*, 2019.

[26] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *CVPR*, 2015.

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

[28] Pavlo Molchanov, Arun Mallya, Stephen Tyree, Iuri Frosio, and Jan Kautz. Importance estimation for neural network pruning. In *CVPR*, 2019.

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

[30] Xuefei Ning, Tianchen Zhao, Wenshuo Li, Peng Lei, Yu Wang, and Huazhong Yang. Dsa: more efficient budgeted pruning via differentiable sparsity allocation. In *ECCV*, 2020.

[31] Prajit Ramachandran, Barret Zoph, and Quoc V. Le. Searching for activation functions. *arXiv preprint arXiv:1710.05941*, 2017.

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

[33] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: inverted residuals and linear bottlenecks. In *CVPR*, 2018.

[34] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In *ICLR*, 2015.

[35] Zi Wang, Chengcheng Li , and Xiangyang Wang. Convolutional neural network pruning with structural redundancy reduction. In *CVPR*, 2021.

[36] Bing Xu, Naiyan Wang, Tianqi Chen, and Mu Li. Empirical evaluation of rectified activations in convolutional network. *arXiv preprint arXiv:1505.00853*, 2015.

[37] Jianbo Ye, Xin Lu, Zhe Lin, and James Z. Wang. Rethinking the smaller-norm-less-informative assumption in channel pruning of convolution layers. In *ICLR*, 2018.

[38] Ruichi Yu, Ang Li, Chun-Fu Chen, Jui-Hsin Lai, and Vlad I. Morariu. Nisp: pruning networks using neuron importance score propagation. In *CVPR*, 2018.

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

[40] Zhuangwei Zhuang, Mingkui Tan, Bohan Zhuang, Jing Liu, Yong Guo, Yong Guo, Junzhou Huang, and Jinhui Zhu. Discrimination-aware channel pruning for deep neural networks. In *NIPS*, 2018.