Constraint-Aware Importance Estimation for Global Filter Pruning under Multiple Resource Constraints

Yu-Cheng Wu[1,2], Chih-Ting Liu[1,2], Bo-Ying Chen[1,2], Shao-Yi Chien[1,2]
[1]NTU IoX Center, National Taiwan University
[2]Graduate Institute of Electronic Engineering, National Taiwan University
{yuchengwu, jackieliu, byc}@media.ee.ntu.edu.tw
sychien@ntu.edu.tw

Abstract

Filter pruning is an efficient way to structurally remove the redundant parameters in convolutional neural network, where at the same time reduces the computation, memory storage and transfer cost. Recent state-of-the-art methods globally estimate the importance of each filter based on its impact to the loss and iteratively remove those with smaller values until the pruned network meets some resource constraints, such as the commonly used number (or ratio) of filter left. However, when there is a more practical constraint like the total number of FLOPs, they ignore its relation to the estimation of filter importance. We propose a novel method called Constraint-Aware Importance Estimation (CAIE) that integrates information of the impact on the given resource into the original importance estimation only based on loss when pruning each filter. Moreover, our CAIE can be generalized to the pruning problem under multiple resource constraints simultaneously. Extensive experiments show that under the same multiple resource constraints, the model pruned with our CAIE method can not only accurately meet the constraints but also achieve the optimal performance results when comparing to existing state-of-the-art methods.

1. Introduction

In recent years, Convolutional Neural Networks (CNN) have been widely adopted in many computer vision tasks, such as image recognition [6], object detection [18] or semantic segmentation [1]. To achieve better performance in these tasks, CNNs are designed to be deeper and more complex, which are accordingly more computation demanding. Although it can be achieved by performing on multiple powerful GPUs, nonetheless, in the real-world applications on mobiles or embedded devices, the limited computation resources and constraints, such as the total number of operations, latency or energy consumption, will hinder the deployment of those CNN models. Thus, how to optimize and accelerate the heavy network but maintain the performance at the same time would be a critical issue to solve.

Network pruning is a common solution for optimizing the model. Given a large and deep neural network, the goal of pruning is trying to obtain an optimal sub-network with acceptable performance drop, or even sometimes resulting in a small gain, by exploring and removing the redundant parts of the model. Based on the categories of a unit being pruned at a time, filter pruning [12], also known as channel pruning [9], is one promising technique which effectively reduces the computation cost by regarding a structural portion, such as a convolutional filter or a channel in output featuremaps, of the model as an unit when performing network pruning. Furthermore, among all the relevant approaches of filter pruning, the global method [17, 14, 16, 23] which determines the redundant filters based on the whole network is usually more popular than those pruning the filters layer-by-layer [15, 9, 25] because globally removing the redundancy is more flexible and time-efficient. In detail, the procedure of global method can be described as follow: the pruning algorithm first repeatedly removed unimportant filters in a given trained CNN until the pruned network satisfied given pruning objectives. Then, the fine-tuning training process will be conducted to retain the performance.As for the pruning objective, commonly it would be the number (or ratio) of filters left, or other computation resource constraints such as floating point operations (FLOPs), number of total parameters, inference latency, and so on.

The core component of global filter pruning is the process to determine the “importance” of each filter and thus iteratively remove the least important one in the whole network. The correctness of the importance estimation will explicitly affect the final performance results. In the past, the L1-norm [12] or sparsity [13] of a filter is used, while recently, most works [17, 16, 23] achieve outstanding results by estimating the importance of a filter based on its impact on the loss when being removed. However, we found that the "pruning objective" at hands is not taken into consideration during importance estimation in those works, which reduces the correlation between the estimated importance of filters and the optimal solution for pruning the network under the specific objective. Molchanov *et al.* [17] has proposed a method to consider the given pruning objective during importance estimation, but the integration method cannot be proved to achieve an optimal result. In addition, when the objective contains multiple resource constraints, which is practical in the real-world scenario, previous methods can only keep pruning the model until it separately matches all constraints. Because they cannot jointly consider all objectives, the pruning results will not meet all constraints accurately at the same time and consequently lead to a worse performance.

To solve the problems mentioned above, we propose a novel method called *Constraint-Aware Importance Estimation (CAIE)* to integrate the information of given resource constraints into the original importance estimation of filters. Given any single resource constraint as in other works, we first need to define the "resource impact" of a filter, which is the normalized amount of resource reduction in the whole network when the specific filter is removed. Then, by mathematical derivation, we can simply combine the original impact based on the loss function and the impact based on the specific constraint to estimate a new constraint-based importance of a filter. Additionally, when we encounter multiple resource constraints, our integration method can be easily *generalized* to the formulation with multi-constraints. With our CAIE, in each pruning iteration, we can remove the filters that will make the pruned sub-network most close to the objective but with least impact on the loss function, which is therefore the optimal solution compared to others. Moreover, when applying our generalized estimation method under multi-constraints, we can thus achieve the best performance over state-of-the-arts and simultaneously meet all the given constraints accurately. We now highlight the contributions of this work:

- We propose a novel method called Constraint-Aware Importance Estimation (CAIE) to estimate the importance of filters in combination with given resource constraint, which can obtain the best results when comparing to those only based on the loss function.
- The proposed method can be easily generalized into the pruning problem under multiple constraints, which is practical to real-world scenarios.
- Under the same amount of resource consumption of the pruned model, we can achieve the state-of-the-art performance results with our proposed CAIE method.

2. Related Work

2.1. Filter Pruning

Network pruning is a common method to obtain a compact network from a large one by removing the redundant parts. Comparing to the traditional weight pruning [5], which only removes the redundant parameter individually, in CNNs, filter pruning can effectively reduce the computation consumption by treating a convolutional filter in the model as the unit for pruning. To determine the redundant filters, some works focus on evaluating the importance of filters in a single layer and remove unimportant ones "layer-by-layer" while others are interested in the global method that evaluate and prune filters based on the whole network.

Layer-wise filter pruning. Among the methods pruning layer-by-layer, some works [12, 7, 13, 8] believe that there is a strong correlation between the importance of a filter and its corresponded parameter-dependent values, such as its L1-norm [12], L2-norm [7], sparsity [13] or the distance to the geometric median of filters in a single layer [8]. On the other hand, some works introduce using training data to yield the criterion for filter removal [15, 9, 25]. Thinet [15] and CP [9] select filters that can minimize the feature reconstruction error layer-by-layer by solving LASSO problem [21]. Moreover, DCP [25] adopt additional discrimination-aware losses to not only guide the selection of redundant filters but also enhance the discrimination ability of features. However, all of the methods mentioned above can only compare filters in the same layer, in other words, cross-layer comparison is not available. Furthermore, layer-wise filter pruning is time-consuming and requires pre-defined pruning ratio for each layer which may reduce the flexibility of the left network and cause a worse performance result. Therefore, we focus on solving filter pruning problem with the global method.

Global filter pruning. To make the estimated values for filter importance be globally comparable, Molchanov *et al.* [17] utilizes layer-wise normalization technique to rescale the original importance score which is generated by Taylor expansion of the impact on the loss function caused by the removal of filters. NISP [24] measures the importance of features in the final response layer then propagates the importance score for each filter in the whole network from the final response layer to the first layer. Some works [14, 22] try to take advantage of batch-normalization (BN) layers [10]. They enforce the sparsity of the scaling factor γ in the BN layer by adding a regularization term in training, and then prune filters depending on a global threshold over the value of γ. Recently, to assess the importance of a filter more accurately, Molchanov *et al.* [16] modifies the Taylor expansion method in [17] so that the additional normalization is not required. Combining Taylor expansion method and the sparsity enforcement technique, GBN [23] introduces "Gate Decorator" and apply it on BN layers for importance estimation and sparsity training. Although they can obtain a promising performance, except the work [17], none of these methods consider the given constraint during importance estimation of filters.

2.2. Constraint-based Network Optimization

Some works try to integrate the information of given constraints when optimizing the network. Molchanov *et al.* [17] add a regularization term about the given constraint to the original score of the filter importance. However, this method introduce an extra parameter λ to control the amount of regularization, which is selected empirically and sensitive to the magnitude of regularization term and the original importance score. LCP [2] adopt evolutionary algorithm to offset the importance of a filter with the given constraints which is originally evaluated by the impact on the loss. Though LCP also consider the given constraints when searching the offset value, their belief that the optimal importance estimation of filters is based on the impact on loss is incorrect since it should also be related to the given constraint. Morphnet [4], a work about network architecture search, introduce the resource-weighted regularizer in the loss function to search the proper width of each layer that is optimal to the targeted resource. Even though, Morphnet can only concern about a single resource at a time. To sum up, our method can easily merge the information of the given constraints during importance estimation in global filter pruning.

3. Proposed Methods

To clearly unify and compare to the works with filter pruning, we will first introduce the preliminaries and define the optimization problem of original filter pruning in the global method. Then, in order to solve the problem under a single constraint, we re-formulate the global filter pruning problem and solve it with our proposed Constraint-Aware Importance Estimation (CAIE), which is then generalized to deal with problems under multiple constraints. At last, we will summarize our iterative pruning and fine-tuning scheme with CAIE in.

3.1. Preliminaries of filter pruning

Given a network with parameters θ and a dataset $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^{N}$ with N training data and label pairs (x_i, y_i), the goal of network training is to minimize the given loss function $\mathcal{L}(\mathcal{D}; \theta)$. In filter pruning, we first define the set of removable filters in the network with parameters θ as $\mathcal{F}(\theta)$. Then, the pruning algorithm will aim to yield a smaller model with left parameters $\theta_{F'}$ that can also minimize the loss function $\mathcal{L}(\mathcal{D}; \theta_{F'})$ by removing a subset of filters $F \subset \mathcal{F}(\theta)$ in the network under a specific constraint, which can be formulated as follow:

$$\underset{F}{\operatorname{argmin}}\, \mathcal{L}(\mathcal{D}; \theta_{F'}) \quad s.t.\, \mathcal{C}(\theta_{F'}) \leq C \tag{1}$$

where $F \cup F' = \mathcal{F}(\theta)$, C is the given pruning constraint and $\mathcal{C}(\theta)$ is the amount of concerned resource consumption for the network with parameters θ. In typical filter pruning problem, the common pruning constraint C is the maximum expected number of filters left, and the corresponded measurement $\mathcal{C}(\cdot)$ would be the total number of filters $|\mathcal{F}(\cdot)|$.

Specifically, when under the setting of pruning in global method, we would first obtain a well-trained network with parameters θ^*. Thus, the objective for the optimization problem can be change into minimizing the difference of performance caused by removing the filters in the network whose parameters are initialized with $\theta = \theta^*$. This difference is commonly evaluated by calculating the *loss impact* [17, 16, 23], defined as $\ell(F)$, when removing the filter set F for θ. The $\ell(F)$ can be formulated as:

$$\ell(F) = \mathcal{M}(\mathcal{L}(\mathcal{D}; \theta),\ \mathcal{L}(\mathcal{D}; \theta_{F'})), \tag{2}$$

where $\mathcal{M}(\cdot)$ is a distance metric function such as squared difference [16] or absolute difference [17, 23]. Therefore, with $\ell(F)$, the optimization problem is re-formulated as:

$$\underset{F}{\operatorname{argmin}}\ \ell(F) \quad s.t.\, \mathcal{C}(\theta_{F'}) \leq C. \tag{3}$$

This problem would then be solved with greedy strategy: iteratively estimating the importance of each filter f, $\mathcal{I}(f)$, in the network left and pruning those least important ones that can minimize the loss impact until the given constraint is satisfied. Commonly, the importance of a filter $\mathcal{I}(f)$ is assigned as its loss impact in this greedy process:

$$\mathcal{I}(f) = \ell(f)\ . \tag{4}$$

During implementation, instead of evaluating $\mathcal{I}(f)$ for all filters in $\mathcal{F}(\theta)$ with $|\mathcal{F}(\theta)|$ pruned models in total, which is time-consuming, $\mathcal{I}(f)$ is usually estimated by first-order Taylor approximation [16], where all required gradients can be obtained by back-propagation at once.

The effectiveness of the criterion "selecting the least important filter" is based on the assumption that the impact of a single filter in the removed filter set can be considered individually, which is only true when a small number of filters are removed. This is why in the solution to problem (3), they iteratively remove part of the most unimportant filters and then re-settle the problem with the network left as a new initialization.

3.2. Single-constraint Importance Estimation

Among previous works, we found that the estimation of filter importance with (4) is lack of information about the given constraint or the concerned resource, which decreases the credibility of acquiring the best pruning result. Hence, we propose a Constraint-Aware Importance Estimation (CAIE) method which aims to combine both the information of constraint and performance simultaneously during importance estimation for a single filter. To better derive our solution, we will first re-formulate the original pruning

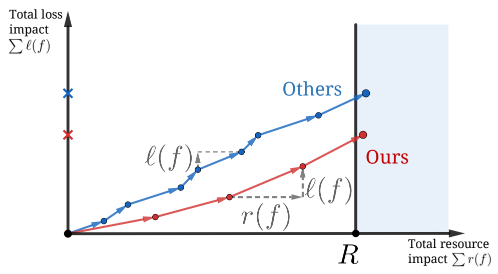


Figure 1: **Comparison of our CAIE to others under the single-constraint pruning problem.** The paths colored in red and blue denote the pruning process generated by our method and others respectively. Each colored vector between two points illustrates the loss impact $\ell(f)$ (vertical component) and the resource impact $r(f)$ (horizontal component) when removing the filter f. Our method which considers the two components jointly can generate a better result with lower total amount of loss impact under the same pruning objective R.

problem (3) into summation of the individual contribution to performance change and resource reduction for a single filter without losing authenticity.

First of all, those common choices of metric function $\mathcal{M}(\cdot)$ ensure the linearity of loss impact, as a result, problem (3) can be rewritten as:

$$\underset{F}{\operatorname{argmin}} \sum_{f \in F} \ell(f) \quad s.t.\ \mathcal{C}(\theta_{F'}) \leq C. \tag{5}$$

In practical usage, the given constraint C could be the maximum value of a certain type of computation resource such as FLOPs. To better solve the pruning problem under such scenarios, we introduce the *resource impact*, $r(F)$, of a filter set F, which is the proportion of the reduction in resource consumption while pruning F:

$$r(F) = \frac{\mathcal{C}(\theta) - \mathcal{C}(\theta_{F'})}{\mathcal{C}(\theta)}\,. \tag{6}$$

Therefore, we can rewrite (5) with our defined resource impact:

$$\underset{F}{\operatorname{argmin}} \sum_{f \in F} \ell(f) \quad s.t.\ r(F) \geq R\,, \tag{7}$$

where the pruning objective $R = \frac{\mathcal{C}(\theta)-C}{\mathcal{C}(\theta)}$ is the minimum proportion of total reduction given C.

Last, since we will resolve problem (7) iterativly during the process of pruning, we can apply an useful assumption when a small number of filters are removed at a time: the resource impact of a filter set F is equal to the sum of resource impact of individual filter f in the set F. Accordingly, we formulate the optimization problem of **single-constraint pruning** as:

$$\underset{F}{\operatorname{argmin}} \sum_{f \in F} \ell(f) \quad s.t.\ \sum_{f \in F} r(f) \geq R\,. \tag{8}$$

In particular, as in previous works, applying the constraint C with the number (or ratio) of filters left is just a special case of (8) with $r(f) = \frac{1}{|\mathcal{F}(\theta)|}$:

$$\underset{F}{\operatorname{argmin}} \sum_{f \in F} \ell(f) \ \ s.t.\ \sum_{f \in F} r(f) = \frac{|F|}{|\mathcal{F}(\theta)|} \geq F_0, \tag{9}$$

where $F_0 = \frac{|\mathcal{F}(\theta)|-C}{|\mathcal{F}(\theta)|}$ is the minimum ratio of filters to be removed to the total number of filters.

Now, given this new optimization problem (8), we want to find the optimal solution through ranking filters by estimating suitable importance score function g, $\mathcal{I}(f) = g(\ell(f), r(f))$, which contains information about the given constraint. Intuitively, $\mathcal{I}(f)$ should possess the following characteristics: 1) If two filters f_1, f_2 have the same value of resource impact, $r(f_1) = r(f_2)$, the importance should be dominated by the corresponding loss impact. 2) If two filters have the same loss impact value, $\ell(f_1) = \ell(f_2)$, the one with larger resource impact should have higher priority of being pruned since removing the filter with higher reduction in resource consumption is more beneficial to the progress of pruning.

To possess the property mentioned above, we propose the Constraint-Aware Importance Estimation (**CAIE**) method under a single constraint, which is a feasible form of importance $\mathcal{I}(f)$ to solve problem (8):

$$\mathcal{I}_{sing}(f) = \frac{\ell(f)}{r(f)}\,. \tag{10}$$

We first qualitatively illustrate the effectiveness of our CAIE in Fig. 1, where the path colored in red is with our method and the two axes represent the total number of loss impact and resource impact. Because our importance estimation $\mathcal{I}_{sing}(f)$ represents the performance drop per unit of resource reduction, comparing to the original $\mathcal{I}(f)$ that only based on $\ell(f)$, greedily selecting those filters with lower $\mathcal{I}_{sing}(f)$ would lead to the smallest performance drop when reaching the needed total amount of resource reduction R.

To confirm the correctness of our CAIE under single constraint (10), we give a more rigorous proof as follow:

Proof. $F_{\mathcal{I}}$ and F^* indicate our solution and the optimal solution respectively. In general, resource impact of a single filter $r(f)$ is far less than the pruning objective R, which implies that we can neglect the difference between total resource reduction $\sum_f r(f)$ and pruning objective R in both solutions, hence, $\sum_{f \in F_{\mathcal{I}}} r(f) = \sum_{f \in F^*} r(f)$.

Let $S_0 := F_{\mathcal{I}} \cap F^*$ and suppose that $F_{\mathcal{I}} \neq F^*$, we have $S_1 := F_{\mathcal{I}} \setminus (S_0) \neq \emptyset$ and $S_2 := F^* \setminus (S_0) \neq \emptyset$. Then,

$$\begin{aligned} \sum_{f \in S_1} r(f) &= \sum_{f \in F_{\mathcal{I}}} r(f) - \sum_{f \in S_0} r(f) \\ &= \sum_{f \in F^*} r(f) - \sum_{f \in S_0} r(f) = \sum_{f \in S_2} r(f)\,. \end{aligned} \tag{11}$$

Based on the facts that $F_{\mathcal{I}} \cap S_2 = \emptyset$, $S_1 \subset F_{\mathcal{I}}$ and the criterion "selecting least importance filter" in the pruning algorithm, we have

$$\max_{f \in S_1} \mathcal{I}(f) \le \max_{f \in F_{\mathcal{I}}} \mathcal{I}(f) \le \min_{f \in S_2} \mathcal{I}(f) \; . \tag{12}$$

Therefore,

$$\begin{aligned} \sum_{f \in S_1} \ell(f) = \sum_{f \in S_1} \frac{\ell(f)}{r(f)} r(f) = \sum_{f \in S_1} \mathcal{I}(f) r(f) \\ \le \max_{f \in S_1} \mathcal{I}(f) \cdot \sum_{f \in S_1} r(f) \\ \le \min_{f \in S_2} \mathcal{I}(f) \cdot \sum_{f \in S_2} r(f) \\ \le \sum_{f \in S_2} \mathcal{I}(f) r(f) \\ = \sum_{f \in S_2} \frac{\ell(f)}{r(f)} r(f) = \sum_{f \in S_2} \ell(f) \; , \end{aligned} \tag{13}$$

and

$$\begin{aligned} \sum_{f \in F_{\mathcal{I}}} \ell(f) = \sum_{f \in S_0} \ell(f) + \sum_{f \in S_1} \ell(f) \\ \le \sum_{f \in S_0} \ell(f) + \sum_{f \in S_2} \ell(f) = \sum_{f \in F^*} \ell(f) \; . \end{aligned} \tag{14}$$

We can see that with (14), the total loss impact of our solution $F_{\mathcal{I}}$ will always be equal to or less than that of the optimal solution F^*. In other words, our solution $F_{\mathcal{I}}$ is thus an optimal solution to the problem of single-constraint pruning (8). □

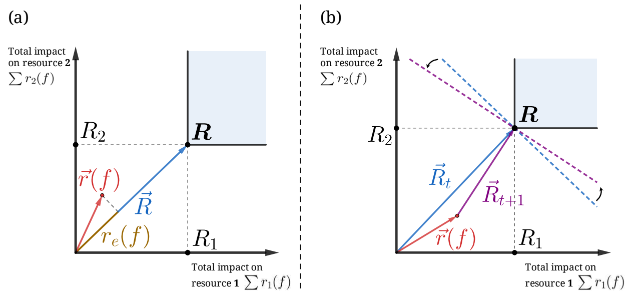
3.3. Multiple-constraint Importance Estimation

In practical scenarios, given any desired platform, we may need to jointly consider some pruning constraints of different resources at the same time, such as regarding the # of parameters left of the model owing to the memory storage and the # of FLOPs based on the platform's computing power. Therefore, we need to generalize the aforementioned single-constraint pruning problem into that under multiple constraints and also generalize the solution with our CAIE.

In formulation, when given k constraints $\{C_i\}_{i=1}^{k}$, we can first generalize (8) to the problem of **multiple-constraint pruning**:

$$\underset{F}{\operatorname{argmin}} \sum_{f \in F} \ell(f) \quad s.t. \; \sum_{f \in F} r_i(f) \ge R_i, \; \forall i \le k \; , \tag{15}$$

with $r_i(f) = \frac{\mathcal{C}_i(\theta) - \mathcal{C}_i(\theta_{f'})}{\mathcal{C}_i(\theta)}$ and the pruning objective $R_i = \frac{\mathcal{C}_i(\theta) - C_i}{\mathcal{C}_i(\theta)}$ for each concerned resource i. Specifically, we can neglect the resource i when $\mathcal{C}_i(\theta) - C_i < 0$, which means its consumption is already lower than the given constraint; thus, the pruning objective of resource i should be modified as $R_i = \min(\frac{\mathcal{C}_i(\theta) - C_i}{\mathcal{C}_i(\theta)}, 0)$.



(a)
Total impact on resource 2 $\sum r_2(f)$
R_2
$\boldsymbol{R}$
$\vec{r}(f)$
$\vec{R}$
$r_e(f)$
R_1
Total impact on resource 1 $\sum r_1(f)$
(b)
Total impact on resource 2 $\sum r_2(f)$
$\boldsymbol{R}$
R_2
$\vec{R}_t$
$\vec{R}_{t+1}$
$\vec{r}(f)$
R_1
Total impact on resource 1 $\sum r_1(f)$

Figure 2: **(a) Illustration of the effective resource impact. (b) The modified problem in multiple-constraint pruning.** We take the problem under two resource constraints (R_1, R_2) as an example and demonstrate the resource impact on the resource plane. (a): The effective resource impact $r_e(f)$ is the scalar projection of $\vec{r}(f)$ to the objective vector $\vec{R}$. (b): Dotted lines are the boundaries of the constraints in the modified problem (19) with vectors $\vec{R}_t$ and $\vec{R}_{t+1}$ in pruning iteration t and $t+1$. As we remove some filters in the network, we will adjust the objective vector from $\vec{R}_t$ to $\vec{R}_{t+1}$ in order to make the pruning direction still point to the point $\boldsymbol{R}$.

To better derive the solution, we need to jointly consider all different resource impacts when removing one filter f. We define the joint resource impact and the overall pruning objective as the vector form, $\vec{r}(f) = \langle r_1(f), r_2(f), ..., r_k(f) \rangle$ and $\vec{R} = \langle R_1, R_2, ..., R_k \rangle$, in the resource space $\mathbb{R}^k$. With the linearity of vectors, the total resource impact $\sum_{f \in F} r_i(f)$ for all resource i when pruning the filter set F can thus be easily obtained by summation of the resource impact vectors $\vec{r}(f)$:

$$\left\langle \sum_{f \in F} r_1(f), \sum_{f \in F} r_2(f), ..., \sum_{f \in F} r_k(f) \right\rangle = \sum_{f \in F} \vec{r}(f) \; . \tag{16}$$

Furthermore, in the space $\mathbb{R}^k$, we found that the direction of $\vec{R}$ is the optimal direction for pruning because the objective point $\boldsymbol{R} = (R_1, R_2, ..., R_k)$ is the closet point on the boundary of the constraints in problem (15) to the origin of the resource space. Hence, when finding the optimal solution, we only need to focus on the components of $r(\vec{f})$ with positive contribution to the direction of $\vec{R}$.

Consequently, we define the ***effective resource impact***, $r_e(f)$, as the scalar projection of $\vec{r}(f)$ onto $\vec{R}$:

$$r_e(f) = \vec{r}(f) \cdot \frac{\vec{R}}{|\vec{R}|} = \frac{\sum_i r_i(f) R_i}{\sqrt{\sum_i R_i^2}} \; , \tag{17}$$

which is also illustrated in Fig. 2 (a). With $r_e(f)$, our *generalized* Constraint-Aware Importance Estimation **(CAIE)** under multiple constraints $\mathcal{I}_{mul}(f)$ can be defined as the

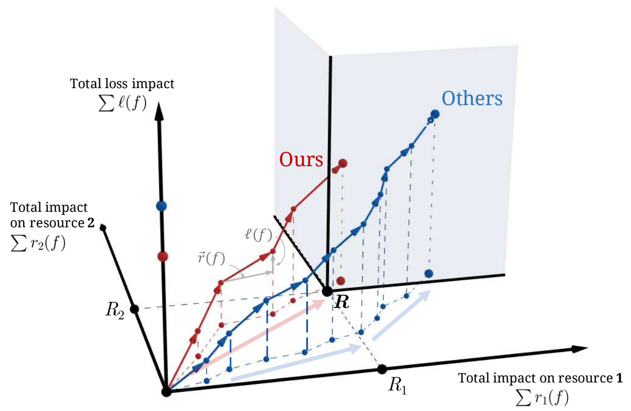


Figure 3: **Comparison of our CAIE to others under the multiple-constraint pruning problem.** We demonstrate the pruning problem under two constraints. The paths colored in red and blue denote the pruning process generated by our method and others respectively. Each colored vector between two points is composed by the loss impact $\ell(f)$ and the resource impact vector $\vec{r}(f)$ when removing the filter f. Our method with CAIE is able to reach the pruning objectives simultaneously and also jointly considers all the impacts to generate a better result with lower total loss impact under the objective $\boldsymbol{R}$.

formula similar to that in single-constraint pruning (10):

$$\mathcal{I}_{mul}(f) = \frac{\ell(f)}{r_e(f)} \,. \tag{18}$$

It's worth noting that in fact, the importance $\mathcal{I}_{mul}(f)$ is the optimal solution to the following modified problem:

$$\operatorname*{argmin}_{F} \sum_{f\in F} \ell(f) \quad s.t. \sum_{f\in F} r_e(f) \geq |\vec{R}| \,, \tag{19}$$

which is transformed from the problem under single constraint (8) with the substitution in some of the notations. Although the boundary of constraints in original problem (15) and that in the modified one (19) are distinctive, shown in Fig. 2 (b), the point $\boldsymbol{R}$ is also the closet point on the boundary in (19) to the origin, which is the same as problem (15). Moreover, from iteration t to $t+1$ among the pruning process, we will also adjust the objective vector $\vec{R}$ whenever we remove a small number of filters. Thus, also shown in Fig. 2 (b), we can always consider the optimal direction in the process and consequently be able to reach the final pruning objective point $\boldsymbol{R}$ accurately.

In Fig. 3, we demonstrate the effectiveness of our CAIE method under multiple constraints (colored in red) when comparing to others only based on the loss impact (colored in blue), where the x-y plane is the resource space and the z-axis denotes the total loss impact. With our CAIE, the path which represents the iterative pruning result always moves toward the pruning objective point $\boldsymbol{R}$ while previous methods can only "separately" match each constraints (i.e. first meet R_1 then R_2). Moreover, since we greedily select the filters with the smallest importance score $\mathcal{I}_{mul}(f)$, which is the slope of performance change on the effective reduction of all the resources, we can acquire the result with smaller overall impact on loss than others when it meets $\boldsymbol{R}$. To summarize, our CAIE method can generate the optimal result under any combination of resource constraints.

Algorithm 1: Global Filter Pruning with CAIE

```
Input: Pre-trained network parameters $\theta^*$, dataset $\mathcal{D}$, $k$
    pruning constraints $\{C_i\}_{i=1}^k$
Output: pruned network parameters $\theta_p^*$
1: Set $\theta^*$ as the initialization of the concerned network $\theta$
2: while $\theta$ not satisfy given constraints do
3:     Estimate the loss impact $\ell(f)$ for each filter f in $\theta$
       with $n$ mini-batches of data in $\mathcal{D}$
4:     Evaluate the pruning objective vector $\vec{R}$ from $\theta$ and
       the resource impact vector $\vec{r}(f)$ for each filter f in $\theta$
5:     Calculate the importance score $\mathcal{I}(f)$ with
       formula (17) and (18)
6:     Remove a filter set $F$ containing $m$ least important
       filters based on $\mathcal{I}(f)$ and acquire a left network $\theta_{F'}$
7:     Set $\theta_{F'}$ as the concerned network $\theta$ for next iteration
8: end while
9: Fine-tuning $\theta$ with $\mathcal{D}$ and yield a pruned model $\theta_p^*$
```

3.4. The Overall Pruning Scheme

Our algorithm of global filter pruning follows the procedure of "iteratively pruning then fine-tuninig". During a single iteration in the pruning stage to remove a small number of filters, we first estimate the loss impact $\ell(f)$ of each filter f following the method proposed in [16]. Next, we will evaluate the resource impact vector $\vec{r}(f)$ and the pruning objective vector $\vec{R}$. With $\ell(f)$, $\vec{r}(f)$ and $\vec{R}$ at hands, our Constraint-Aware Importance Estimation (CAIE) can integrate those components with formula (17) and (18) to generate the importance $\mathcal{I}(f)$ of each filter as the criterion for pruning. The iterative pruning process will continue until the pruned network satisfies the given constraints, then followed by the fine-tuning process to further boost the performance at last. The overall procedure of our proposed algorithm is shown in Alg. 1.

4. Experiments

In this section, we will explain the implementation details of our experiments and demonstrate the effectiveness of our CAIE method.

Table 1: **Ablation Studies of our CAIE.** Each block spaced apart by double line represents a group of experiments. The column "w/ $-$ w/o CAIE" illustrates the performance gain after applying our CAIE comparing to the first row in each block.

Model	**Constraints**	**w/ CAIE**	FLOPs left (%)	Param. left (%)	P. Top-1 (%)	Top-1↓ (%)	w/ $-$ w/o CAIE (%)
ImageNet [19]							
ResNet-50 (orig. top-1 : 76.13%)	$f_{.33}$, $p_{.31}$	✗	**32.83**	**25.94**	71.57	4.56	-
	$f_{.33}$	✓	32.95	49.40	73.90	2.23	2.33
	$p_{.26}$	✓	46.64	25.80	71.96	4.17	0.39
	$f_{.33}$, $p_{.31}$	✓	32.90	30.76	72.39	3.74	0.82
	$f_{.33}$, $p_{.26}$	✓	**32.47**	**25.89**	71.92	4.22	**0.34**
ResNet-50 (orig. top-1 : 76.13%)	$f_{.65}$, $p_{.70}$	✗	**64.83**	**64.27**	75.59	0.54	-
	$f_{.65}$	✓	64.58	85.72	76.02	0.11	0.43
	$p_{.65}$	✓	79.80	64.70	75.80	0.33	0.21
	$f_{.65}$, $p_{.70}$	✓	64.95	69.88	75.83	0.30	0.24
	$f_{.65}$, $p_{.65}$	✓	**64.81**	**64.61**	75.69	0.44	**0.10**
ResNet-34 (orig. top-1 : 73.31%)	$f_{.78}$, $p_{.79}$	✗	**77.55**	**71.43**	72.67	0.64	-
	$f_{.78}$	✓	77.47	90.43	73.15	0.16	0.48
	$p_{.72}$	✓	85.89	71.29	72.72	0.59	0.05
	$f_{.78}$, $p_{.79}$	✓	77.43	78.94	72.91	0.40	0.24
	$f_{.78}$, $p_{.72}$	✓	**77.72**	**71.32**	72.73	0.58	**0.06**
CIFAR-10 [11]							
VGG16-BN (orig. top-1 : 93.34%)	$f_{.44}$, $p_{.20}$	✗	**43.32**	**9.93**	92.94	0.40	-
	$f_{.44}$	✓	44.00	12.55	93.06	0.28	0.12
	$p_{.10}$	✓	42.90	9.69	93.02	0.32	0.08
	$f_{.44}$, $p_{.20}$	✓	43.07	12.19	93.11	0.23	0.17
	$f_{.44}$, $p_{.10}$	✓	**42.43**	**9.89**	92.98	0.36	**0.04**
ResNet-34 (orig. top-1 : 94.13%)	$f_{.40}$, $p_{.15}$	✗	**29.90**	**14.48**	93.34	0.79	-
	$f_{.30}$	✓	29.82	19.95	93.48	0.65	0.14
	$p_{.15}$	✓	35.69	14.79	93.46	0.67	0.12
	$f_{.40}$, $p_{.15}$	✓	35.10	14.88	93.50	0.63	0.16
	$f_{.30}$, $p_{.15}$	✓	**29.64**	**14.79**	93.40	0.73	**0.06**

4.1. Implementation details

Dataset Our CAIE is evaluated on the CIFAR-10 [11] and ImageNet ILSVRC-12 [19]. The CIFAR-10 dataset contains 50k training images and 10k test images in 10 classes, while ImageNet dataset contains 1.28M training images and 50k test images in 1000 classes.

Loss impacts We choose the method proposed by Molchanov *et al.* [16] to evaluate the loss impact $\ell(f)$ of a filter f. The formula of loss impact is $\ell(f) = (\gamma_c \frac{\partial \mathcal{L}}{\partial \gamma_c} + \beta_c \frac{\partial \mathcal{L}}{\partial \beta_c})^2$, where γ_c and β_c are the scaling and shifting parameters of the following BN layer in the channel c corresponding to the filter f. The gradients $\frac{\partial \mathcal{L}}{\partial \gamma_c}$ and $\frac{\partial \mathcal{L}}{\partial \beta_c}$ in the formula above would be computed whenever a mini-batch of training data is given. We average the loss impacts calculated in n mini-batches using running average with coefficient 0.9. Additionally, the gradients would also be utilized in updating network when evaluating the loss impacts of filters. If the network such as ResNet [6] contains residual blocks that some specific layers should be pruned in the same way, as described in [16], we will sum over the corresponding loss impacts in the same channel of these grouped layers as the overall loss impact.

Resource impacts The i^{th} resource impact $r_i(f)$ of the filter f among the same layer share the same value. Thus, we can evaluate the resource impact for layer l by randomly removing a filter in layer l plus the corresponding channel of each filter in the succeeding $(l+1)^{th}$ layer then measuring the proportion of reduction in the concerned resources. Also, when we encounter residual blocks containing grouped layers, we will randomly remove an output channel in these layers following the rule in the works [3, 23] to evaluate the overall resource impact.

Pruning and fine-tuning In both stages, the batch size is set to be 256 and 64 in CIFAR-10 and ImageNet respectively. In a single pruning iteration, the number of batches n used for estimating importance and the number of pruned filters m are set to be 30 and 25 in both datasets. For optimizing neural network, we use SGD with initial learning rate 10^{-3} and runs for 30 epochs in ImageNet and 240 epochs in CIFAR-10 of the fine-tuning stage. The learning rate will decay by 5 every 10 epochs in ImageNet and every 80 epochs in CIFAR-10.

Table 2: **Comparison to state-of-the-arts on ImageNet.** To compared with others, we set the resource constraints based on the resource left of the pruned model in other works.

Model	Orig. Top-1 (%)	Method	FLOPs left (%)	Param. left (%)	P. Top-1 (%)	Top-1$\downarrow$ (%)
ResNet-50	76.18	Taylor-FO-BN-56% [16]	32.76	30.86	71.69	4.49
	76.13	**Ours** ($f_{.33},\ p_{.31}$)	32.90	30.76	**72.39**	**3.74**
ResNet-50	72.88	Thinet-30 [15]	34.66	28.49	68.42	4.46
	76.13	**Ours** ($f_{.33},\ p_{.26}$)	32.47	25.89	**71.92**	**4.22**
ResNet-50	76.15	FPGM-only 30% [8]	58.80	-	75.59	0.56
	76.13	**Ours** ($f_{.55}$)	54.77	77.35	**75.62**	**0.53**
ResNet-50	76.18	Taylor-FO-BN-81% [16]	65.03	69.92	75.48	0.70
	76.13	**Ours** ($f_{.65},\ p_{.70}$)	64.95	69.88	**75.83**	**0.30**
ResNet-50	-	NISP-50-B [24]	55.99	56.18	-	0.89
	76.13	**Ours** ($f_{.56}, f_{.56}$)	55.89	55.84	**75.25**	**0.88**
ResNet-34	73.31	Taylor-FO-BN-82% [16]	77.74	78.90	72.83	0.48
	73.23	Li *et al.* [12]	75.80	89.20	72.17	1.04
	73.31	**Ours** ($f_{.78},\ p_{.79}$)	77.43	78.94	**72.91**	**0.40**

4.2. Evaluation

We conduct experiments in Table 1 and 2 to verify our method when given pre-trained models, such as the ResNet series [6] and the VGG network [20], and some resource constraints. In our experiments, we adopt two commonly used concerned resources, which is the total FLOPs (f) and the network parameters (p) respectively. The given constraints would be the maximum proportion of the specific resource could remain. For example, ($f_{.33},\ p_{.31}$) indicates that there should be at most 33% FOLPs and 31% parameters left of the pruned model. The performance of a pruned model would then be evaluated by its top-1 accuracy (P. Top-1) and the accuracy drop after pruning (Top-1$\downarrow$).

Effectiveness of our CAIE We conduct ablation studies in Table 1 to compare the pruning results of the model **with** and **without** applying our CAIE on ImageNet and CIFAR-10. Each block in the table containing five rows shows a group of experiments on the pre-trained model and its original top-1 accuracy. The first row in each block is the "baseline" result without CAIE, which only takes the loss impact as the importance while pruning until separately meets the constraints. The second and third row are used to confirm the correctness of our CAIE under a single constraint. As we can see, the results in these two rows have the same amount of concerned resource compared to the pruned result of baseline yet reach better performance. The fourth row demonstrates the flexibility of our CAIE that can accurately adapt to the given multiple constraints. In some cases ($f_{.44},\ p_{.20}$ in VGG16-BN), the result generated by CAIE may not meet the constraints simultaneously, but the performance still be better than that in the baseline result. Last, the fifth row is to affirm the effect of CAIE when the given multiple constraints is set to the same as the baseline results, such as ($f_{.33},\ p_{.26}$) is the same as (32.83, 25.94) in the first block. As we can see, our results can still outperform the baseline for all the models. Altogether, our CAIE can always yield improvement in performance when given any constraints.

Comparison to state-of-the-arts In Table 2, we compare our CAIE with others on the ImageNet. Given a pruning result in other works, we will conduct experiments with our CAIE under the resource constraints corresponding to the resources left of others. Compared to the state-of-the-art Taylor-FO-BN [16], which contains the same calculation of loss impact and the pruning procedure as ours, results with CAIE can achieve better performance. Furthermore, when comparing to those with different importance estimation and the pruning process [15, 8, 24, 12], our method still can obtain the best results. It is worth noting that for fair comparison, we did not show the results of GBN [23] because they apply other losses to reinforce the sparsity when training.

5. Conclusions

In this work, we propose a novel method called Constraint-Aware Importance Estimation (CAIE) to estimate the importance of filters in the network under the given multiple resource constraints, which integrates information of the impact on the considered resources with the impact on loss function when removing a filter. We demonstrate the effectiveness of our method and we can achieve state-of-the-art performance comparing to others under the same amount of resource consumption of the pruned model.

Acknowledgment

This research was supported in part by the Ministry of Science and Technology of Taiwan (MOST 108-2633-E-002-001), National Taiwan University (NTU-108L104039), Intel Corporation, Delta Electronics and Compal Electronics.

References

[1] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 40(4):834–848, 2017.

[2] Ting-Wu Chin, Cha Zhang, and Diana Marculescu. Layer-compensated pruning for resource-constrained convolutional neural networks. *arXiv preprint arXiv:1810.00518*, 2018.

[3] Xiaohan Ding, Guiguang Ding, Yuchen Guo, and Jungong Han. Centripetal sgd for pruning very deep convolutional networks with complicated structure. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4943–4953, 2019.

[4] Ariel Gordon, Elad Eban, Ofir Nachum, Bo Chen, Hao Wu, Tien-Ju Yang, and Edward Choi. Morphnet: Fast & simple resource-constrained structure learning of deep networks. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1586–1595, 2018.

[5] 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 (NeurIPS)*, pages 1135–1143, 2015.

[6] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016.

[7] 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.

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

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

[10] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.

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

[12] Hao Li, Asim Kadav, Igor Durdanovic, Hanan Samet, and Hans Peter Graf. Pruning filters for efficient convnets. *arXiv preprint arXiv:1608.08710*, 2016.

[13] Chih-Ting Liu, Yi-Heng Wu, Yu-Sheng Lin, and Shao-Yi Chien. Computation-performance optimization of convolutional neural networks with redundant kernel removal. In *2018 IEEE International Symposium on Circuits and Systems (ISCAS)*, pages 1–5. IEEE, 2018.

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

[15] Jian-Hao Luo, Jianxin Wu, and Weiyao Lin. Thinet: A filter level pruning method for deep neural network compression. In *IEEE International Conference on Computer Vision (ICCV)*, pages 5058–5066, 2017.

[16] Pavlo Molchanov, Arun Mallya, Stephen Tyree, Iuri Frosio, and Jan Kautz. Importance estimation for neural network pruning. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 11264–11272, 2019.

[17] 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.

[18] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 91–99, 2015.

[19] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *International journal of computer vision*, 115(3):211–252, 2015.

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

[21] Robert Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society: Series B (Methodological)*, 58(1):267–288, 1996.

[22] Jianbo Ye, Xin Lu, Zhe Lin, and James Z Wang. Rethinking the smaller-norm-less-informative assumption in channel pruning of convolution layers. *arXiv preprint arXiv:1802.00124*, 2018.

[23] Zhonghui You, Kun Yan, Jinmian Ye, Meng Ma, and Ping Wang. Gate decorator: Global filter pruning method for accelerating deep convolutional neural networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 2130–2141, 2019.

[24] Ruichi Yu, Ang Li, Chun-Fu Chen, Jui-Hsin Lai, Vlad I Morariu, Xintong Han, Mingfei Gao, Ching-Yung Lin, and Larry S Davis. Nisp: Pruning networks using neuron importance score propagation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 9194–9203, 2018.

[25] 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 (NeurIPS)*, pages 875–886, 2018.