CAP: instance complexity-aware network pruning

Jiapeng Wang 1, Ming Ma 1 and Zhenhua Yu 1,2*

1School of Information Engineering, Ningxia University, Yinchuan, 750021, China
2Collaborative Innovation Center for Ningxia Big Data and Artificial Intelligence co-founded by Ningxia Municipality and Ministry of Education, Yinchuan, 750021, China

Abstract

Existing differentiable channel pruning methods often attach scaling factors or masks behind channels to prune filters with less importance, and assume uniform contribution of input samples to filter importance. Specifically, the effects of instance complexity on pruning performance are not yet fully investigated. In this paper, we propose a simple yet effective differentiable network pruning method CAP based on instance complexity-aware filter importance scores. We define instance complexity related weight for each sample by giving higher weights to hard samples, and measure the weighted sum of sample-specific soft masks to model non-uniform contribution of different inputs, which encourages hard samples to dominate the pruning process and the model performance to be well preserved. In addition, we introduce a new regularizer to encourage polarization of the masks, such that a sweet spot can be easily found to identify the filters to be pruned. Performance evaluations on various network architectures and datasets demonstrate CAP has advantages over the state-of-the-arts in pruning large networks. For instance, CAP improves the accuracy of ResNet56 on CIFAR-10 dataset by 0.33% after removing 65.64% FLOPs, and prunes 87.75% FLOPs of ResNet50 on ImageNet dataset with only 0.89% Top-1 accuracy loss.

1 Introduction

Convolutional neural networks have shown excellent performance in computer vision tasks [1, 2] such as object detection and image recognition. To make the network perform well in various tasks, the size of network structure has been continuously increased, which results in extensive computation consumption. Deploying such large networks to edge devices is impossible due to limited computing resources. To cope with this issue, model compression has been introduced to obtain compact and efficient subnetworks with little damage to the model performance.

Structured network pruning is a widely used model compression technique, it aims to find and remove redundant structures from a pre-trained or baseline network. Conventional static pruning methods [3, 4, 5, 6, 7, 8, 9] often introduce channel-wise scaling factors or masks to define the importance of the filters, and employ sparsity constraints such as \(\ell_1\) regularizer to push some of the scaling factors towards zero. For instance, Zhuang et al. [10] employs a polarization regularizer to find redundant filters. In general, a threshold on the scaling factors is explored to separate the filters into two parts for removing the less important one, and fine-tuning of the pruned network is required to compensate the accuracy loss. However, if there are no suitable spots that completely separate important filters from redundant ones, the model accuracy of the pruned network may be heavily affected. Therefore, clearly dividing the filters into two parts and maximizing the gap between them are essential for preserving the model capability.

*Correspondence: zhyu@nxu.edu.cn

Preprint. Under review.
Figure 1: Uniform weights used in existing methods, and non-uniform weights proposed in CAP to measure the filter importance. CAP gives higher weights to complex samples that are difficult to classify.

The complexity of input samples may also have considerable influence on network pruning. Intuitively, classification of hard samples may require more complex network structure to make an accurate prediction, and the importance of a filter is more likely related to hard samples. However, existing pruning methods often assume a uniform contribution of different samples during network pruning, and not yet fully investigate the effects of instance complexity on filter importance. Incorporating such information into pruning process may be helpful to yield better pruning results.

In this paper, we propose an instance Complexity-Aware network Pruning (CAP) method to prune a large pre-trained (baseline) network. Similar to related approaches, CAP represents the importance of each filter with a soft mask with value between 0 and 1, and removes the filters with masks close to zero. Our method also shows two new features when compared to the existing methods: 1) to model the non-uniform contribution of different samples to filter importance, we define the importance scores of all filters with a weighted sum of sample-specific soft masks, where the weight of each sample is closely related to the complexity of the sample and higher weights are given to hard samples (as shown in Figure 1); 2) to clearly separate the important filters from redundant ones, we introduce a new regularizer on the soft masks of filters to encourage polarization of the masks, such that a sweet spot can be easily found to divide the filters into two parts with a large margin. These specific modeling aspects enable CAP to yield better pruning results than the state-of-the-art (SOTA) methods. An illustration of CAP framework is given in Figure 2.

Our main contributions are threefold:

- By fully exploiting the difference of instance complexity, non-uniform contribution of input samples to filter importance is introduced to give higher weights for hard samples, which reduces the performance gap between the pruned and original networks.
- By employing a new regularizer on the soft masks of filters, the masks of important filters are pushed towards 1 and those of redundant filters are pushed towards 0, thus a sweet spot can be easily found to separate the two parts of filters.
- Extensive experiments on multiple datasets and popular network architectures demonstrate CAP outperforms the SOTAs by pruning more FLOPs with less accuracy loss.

2 Related Works

2.1 Network pruning

Network pruning methods can be divided into unstructured and structured pruning. Unstructured pruning [11, 12, 13] often exploits sparsity-oriented penalty on network weights to remove unimportant elements in weight tensors, and is not hardware friendly since it requires specially designed library for accelerating multiplication between sparse matrices. By comparison, structured pruning [14, 15, 16, 17] removes regular structures such as channels, filters, neurons and layers of network under specific pruning criteria. Early works mainly adopt an iterative process to generate and evaluate
Figure 2: An illustration of CAP framework. CAP measures the weighted sum of sample-specific soft masks to represent the filter importance, and use the weighted soft masks to obtain output feature maps of the pruned network. The sample-specific soft masks are learned by passing baseline features of each sample through a fully connected network (Mask network), and the weight of each sample is calculated with cross entropy loss. The model parameters are jointly optimized by minimizing MSE loss between baseline features and pruned features, and sparsity loss related to the weighted soft masks.

2.2 Neural architecture search

Neural Architecture Search (NAS) has attracted more attention for its capability to automatically explore a high-performance architecture from a large search space. Early NAS works are based on either reinforcement learning [26, 27] or revolutionary algorithm [28, 29]. To continuously deal with the search space, differentiable architecture search [30] is proposed to facilitate optimization techniques such as gradient descent to be employed for finding the optimal architecture. Our method can also be treated as a NAS process where the channel-wise masks form the search space. Compared with existing NAS methods, our work fully considers the non-uniform contribution of samples to filter importance.
3 Proposed CAP Framework

3.1 Notations and Preliminaries

Given a training dataset \( \{x_i, y_i\}_{i=1}^N \), we aim to get a well performed pruned network \( \mathcal{P} \) from the baseline network \( \mathcal{B} \). Here \( x_i \) denotes input image and \( y_i \) is a one-hot encoding vector to represent the label of \( x_i \). The pruned network \( \mathcal{P} \) has same structure to \( \mathcal{B} \). For channel pruning, the output of each filter in \( \mathcal{P} \) is multiplied by a soft mask before fed into next layer, and all masks are jointly optimized to find and remove filters with mask close to zero. Suppose there are a total of \( n \) filters in \( \mathcal{P} \), the feature map from the \( k \)-th filter is represented by \( F_k \in \mathbb{R}^{W_k \times H_k} \) (\( W_k \) is the height and \( H_k \) is the width of the feature map), and the channel-wise masks are denoted by a \( n \)-dimensional vector \( m \) of which the \( k \)-th element is in \([0, 1]\) and used to scale \( F_k \). Here \( m \) is treated as importance scores of the filters. For each input image \( x_i \), the feature map outputted by the last FC layer is denoted by \( \hat{y}_i = f(x_i; W_b) \) for the baseline network and \( \hat{g}_i = f(x_i; m; W_p) \) for the pruned network. Here \( W_b \) and \( W_p \) refer to the baseline network and the pruned network, respectively. Network pruning can be formulated as the following optimization problem:

\[
\min_{m, W_p} \sum_{i=1}^N \mathcal{L}_{mse}(\hat{y}_i, \hat{g}_i) + \lambda_1 R(W_p) + \lambda_2 R(m)
\]

where \( \mathcal{L}_{mse} \) denotes mean squared error (MSE), \( R(W_p) \) is a regularizer (e.g. \( \ell_2 \)-norm) on weights of the pruned network, \( R(m) \) represents a sparsity regularizer on \( m \), coefficients \( \lambda_1 \) and \( \lambda_2 \) control the weights of \( R(W_p) \) and \( R(m) \), respectively. A larger \( \lambda_2 \) induces sparser masks \( m \) and then yields a more compact pruned network \( \mathcal{P} \).

3.2 Definition of filter importance

Motivated by the concept of instance-aware pruning employed in dynamic network pruning [31, 9], we assume each input sample \( x_i \) characterizes a distribution of filter importance. We use \( m_i = g(x_i) \) to denote sample-specific soft masks defined by \( x_i \), here \( m_i \) is a \( n \)-dimensional vector. Specifically, we model \( m_i \) with a fully connected network \( \mathcal{M} \) consisting of three FC layers and a sigmoid layer. The fully connected layer plays the role of "classifier" in the convolutional neural network, and the FC layer maps the learned feature representation to the tag space of the sample. Only relying on the output of the baseline network \( \hat{y}_i \) cannot meet our purpose. Therefore, instead of directly using \( x_i \) as input, we fed the baseline feature \( \hat{y}_i \) into \( \mathcal{M} \) to output the masks \( m_i \), i.e. \( m_i = g(\hat{y}_i; W_m) \). \( W_m \) means the mask network. This is reasonable as \( \hat{y}_i \) is the high-level feature representation of \( x_i \). The importance scores of filters are then defined as the weighted sum of sample-specific soft masks:

\[
m = \sum_{i=1}^N \alpha_i g(\hat{y}_i; W_m) = \sum_{i=1}^N \alpha_i g(f(x_i; W_b); W_m)
\]

where \( \alpha_i \) denotes the weight of the \( i \)-th sample, and is set to \( \frac{1}{N} \) when assuming a uniform contribution. To consider non-uniform complexity of input samples, we propose to give higher weights to hard samples that are more difficult to classify. Following the approach adopted in [9], we use cross entropy (CE) loss \( L_{ce} \) to define the complexity of each sample, and calculate \( \alpha_i \) as follows:

\[
\alpha_i = \frac{L_{ce}(y_i, \hat{y}_i)}{\sum_j L_{ce}(y_j, \hat{y}_j)}
\]

Given above definitions, the inferred \( m \) in Eq.2 is a weighted average of instance-aware soft masks, and used to scale the output features of filters for all samples.

3.3 Regularizer on soft masks

To determine which filters should be pruned, we need to push the soft masks of redundant filters towards 0 and those of important filters towards 1 to better preserve model capability after pruning. \( \ell_1 \)-norm has been widely used as a sparsity regularizer [32, 9], while it may not promise finding of sweet spots that clearly separate filters into two parts. To cope with this problem, we introduce a new
regularizer on the soft masks of filters by following a similar idea in \cite{11}:

\[ R(m) = \sum_{k=1}^{n} m_k + t(1 - \text{var}(m)) \]  

(4)

where \( \text{var}(m) \) denotes the variance of \( m_1, m_2, \ldots, m_n \). The first term is the \( \ell_1 \)-norm which reaches its minimum when all the \( m_i \) are zero, and minimizing the second term \( 1 - \text{var}(m) \) enables \( m_i, 1 \leq i \leq n \) be far away from the mean. Combining these two terms makes it possible to push the soft masks of some filters to 0 and those of remaining filters to 1, such that important filters can be easily distinguished from redundant ones. The parameter \( t \) is introduced to balance the two terms.

### 3.4 Optimization

The optimization of the mask network \( \mathcal{M} \) is done in an end-to-end fashion. When the mask network is optimized, it means that the mask generated by the mask network is also changing. By combining the MSE loss and regularizers on \( W_p, m \) and \( W_m \), we obtain the following objective function that can be optimized under a differentiable manner:

\[
\min_{W_p, W_m} \sum_{i=1}^{N} \mathcal{L}_{mse}(\hat{y}_i, \hat{g}_i) + \lambda_1 \| W_p \|_2 + \lambda_2 \| W_m \|_2 \\
+ \lambda_3 \| m \|_1 + \lambda_4 (1 - \text{var}(m))
\]  

(5)

where \( \lambda_1 \) and \( \lambda_2 \) are set to 5e-4, coefficients \( \lambda_3 \) and \( \lambda_4 \) are selected according to the desired FLOPs reduction rate in different experiments. The function can be optimized using conventional mini-batch gradient descent algorithm.

### 3.5 Pruning Strategy

After model training is completed, we obtain the soft masks \( m \) for each batch of inputs, and calculate the average of \( m \) among all batches as the final soft masks. We prune the filters whose soft masks are close to 0, and still need to find a threshold to identify these redundant filters. In \cite{11}, the threshold is selected by scanning the bimodal distributed scaling factors. Similar to this approach, in our method the filters are clearly separated into two parts one of which locates near to 0 and another locates near to 1 due to the polarization effect, therefore we do not need to explore a threshold on the mask values and simply use 0.5 as a split spot to divide the filters into two groups. Bases on this pruning strategy, the filters with soft masks near to 0 can be automatically removed. After pruning, the network is fine-tuned on training data.

### 4 Experiments

#### 4.1 Experiment Setting

We evaluated our method on CIFAR-10 \cite{33}, CIFAR-100 \cite{33} and ImageNet \cite{34}, respectively. CIFAR-10 dataset contains 60k images with size of \( 32 \times 32 \) for 10 classes, we used 50k images as training set and remaining 10k images as test set. CIFAR-100 dataset has 100 classes with 500 training images and 100 test images per class. The large-scale dataset ImageNet contains 1k classes, including 1.28 million images as training set and 50k images as verification set. On CIFAR-10 dataset, we evaluated proposed method on ResNet32, ResNet56 and ResNet110 models \cite{35}. On CIFAR-100 dataset, we evaluated our method on ResNet32 and ResNet56 models. On ImageNet dataset, we assessed our proposed method on ResNet50.

**Implementation Details:** The baseline training was set to run 200 epochs on CIFAR-10, CIFAR-100 and ImageNet. The initial learning rate was set to 0.1, and batch size was set to 128. The learning rate was multiplied by 0.1 after every 30 epochs. We used the mini-batch gradient descent method for optimization with momentum of 0.9 and weight decay of 0.0004. In the pruning stage, for different baseline, we train on different datasets for 30 epochs with a learning rate of 0.01 and multiply the learning rate by 0.1 after every 5 epochs. The pruned networks were fine-tuned with same parameters as used in training baselines, except that the initial learning rate was set to 0.01. All experiments were conducted on one RTX TITAN GPU by PyTorch.
Table 1: Compare the results of ResNet32, ResNet56 and ResNet110 on CIFAR-10 dataset. ACC↓ represents the percent of accuracy drop after pruning. Flops↓ represents FLOPs reduction rate. Best results are bolded.

| Network  | Method   | Baseline Acc(%) | Pruned Acc(%) | Acc↓ (%) | Flops↓ (%) |
|----------|----------|-----------------|---------------|----------|------------|
| ResNet32 | FPGM [38] | 92.63           | 92.82         | -0.19    | 53.20      |
|          | LFPC [36] | 92.63           | 92.12         | 0.51     | 52.60      |
|          | Wang et al. [42] | 93.18        | 93.27         | -0.09    | 49.00      |
|          | **Ours**  | 92.94           | 93.79         | **-0.85** | **55.54**  |
|          | MainDP [9] | **92.66**       | **92.15**     | **-0.51** | **63.20**  |
|          | LRF [37]  | 92.49           | 92.54         | -0.05    | 62.00      |
|          | **Ours**  | 92.94           | 93.72         | **-0.78** | **62.42**  |
| ResNet56 | FPGM [38] | 93.59           | 93.26         | 0.33     | 52.60      |
|          | HRank [21] | 93.26           | 93.17         | 0.09     | 50.00      |
|          | LFPC [36] | 93.59           | 93.34         | 0.25     | 52.90      |
|          | DMC [6]   | 93.62           | 93.69         | -0.07    | 50.00      |
|          | SRR-GR [39] | 93.38        | 93.75         | -0.37    | 53.80      |
|          | SCP [8]   | 93.69           | 93.23         | 0.46     | 51.50      |
|          | DPFPS [25] | 93.81           | 93.20         | 0.61     | 52.86      |
|          | Wang et al. [42] | 93.69        | 93.76         | -0.07    | 50.00      |
|          | **Ours**  | 93.25           | 93.67         | **-0.42** | **56.06**  |
|          | LRF [37]  | 93.45           | 93.73         | **-0.28** | **62.40**  |
|          | MainDP [9] | 93.70           | 93.64         | 0.06     | 62.40      |
|          | **Ours**  | 93.25           | 93.58         | **-0.33** | **65.64**  |
| ResNet110| FPGM [38] | 93.68           | 93.85         | -0.17    | 52.30      |
|          | TAS [40]  | 94.97           | 94.33         | 0.64     | 53.00      |
|          | **Ours**  | 93.60           | 93.99         | **-0.39** | **53.18**  |
|          | LRF [37]  | 93.76           | 94.34         | **-0.58** | **62.60**  |
|          | HRank [21] | 93.50           | 92.65         | 0.85     | 68.60      |
|          | LFPC [36] | 93.68           | 93.79         | -0.11    | 60.30      |
|          | **Ours**  | 93.60           | 93.85         | -0.25    | **68.81**  |

Table 2: Compare the results of ResNet32 and ResNet56 on CIFAR-100 dataset. ACC↓ represents the percent of accuracy drop after pruning. Flops↓ represents FLOPs reduction rate.

| Network  | Method   | Baseline Acc(%) | Pruned Acc(%) | Acc↓ (%) | Flops↓ (%) |
|----------|----------|-----------------|---------------|----------|------------|
| ResNet32 | LCCL [41] | 70.08           | 67.39         | 2.69     | 37.5       |
|          | SFP [8]  | 69.77           | 68.37         | 1.40     | 41.5       |
|          | FPGM [38] | 69.77           | 68.52         | 1.25     | 41.5       |
|          | TAS [40]  | 70.6            | 66.94         | 3.66     | 41         |
|          | **Ours**  | 69.51           | 70.49         | **-0.98** | **51.57**  |
| ResNet56 | LCCL [41] | 71.33           | 67.39         | 2.96     | 39.3       |
|          | SFP [8]  | 71.4            | 68.79         | 2.61     | 52.6       |
|          | FPGM [38] | 71.41           | 69.66         | 1.75     | 52.6       |
|          | TAS [40]  | 73.18           | 72.25         | 0.93     | 51.3       |
|          | **Ours**  | 71.34           | 71.93         | **-0.59** | **56.06**  |

**Hyperparameter:** During pruning process, we set two hyper-parameters $\lambda_3$ and $\lambda_4$ to achieve target FLOPs reduction rate. In our experiments, we set $\lambda_3$ to 0.001 and $\lambda_4$ to a value in the range of [1, 20] on CIFAR-10 and CIFAR-100. We set $\lambda_3$ to 0.001 and explore $\lambda_4$ in [20, 80] for target FLOPs reduction on ImageNet.

4.2 Results on CIFAR-10 and CIFAR-100

We first evaluated our proposed method on CIFAR-10 dataset by pruning ResNet-32, ResNet-56 and ResNet-110 networks. As shown in Table 1, we compared CAP to some recently proposed
Table 3: Compare the results of ResNet50 on ImageNet dataset. Top-1 and Top-5 accuracy denote the different accuracy of baseline. Top-1 pruned and Top-5 pruned accuracy represent the accuracy after the pruning. Top-1↓ and Top-5↓ represent the percent of accuracy drop after pruning. FLOPs↓ represents FLOPs reduction rate.

| Network   | Method | Top-1 Acc(%) | Top-1 Pruned Acc(%) | Top-1↓ (%) | Top-5 Acc(%) | Top-5 Pruned Acc(%) | Top-5↓ (%) | FLOPs↓ (%) |
|-----------|--------|--------------|---------------------|------------|--------------|---------------------|------------|------------|
| ResNet50  | OTO [17]| 76.10        | 74.70               | 1.4        | 92.90        | 92.10               | 0.8        | 64.50      |
|           | OTO* [17]| 76.10        | 75.10               | 1.00       | 92.90        | 92.50               | 0.4        | 64.50      |
|           | CHIP [22]| 76.15        | 75.26               | 0.89       | 92.87        | 92.53               | 0.34       | 62.80      |
|           | ResRep [43]| 76.15        | 75.30               | 0.85       | 92.87        | 92.47               | 0.4        | 62.10      |
|           | CHIP [22]| 76.15        | 71.98               | 4.17       | 92.87        | 91.01               | 1.86       | 62.10      |
|           | Liu et al. [44]| 76.79        | 73.94               | 2.85       | -            | -                   | -          | 75.00      |
|           | DMCP [24]| 76.60        | 74.40               | 2.2        | -            | -                   | -          | 73.17      |
|           | Zhuang et al. [10]| 76.15        | 69.10               | 7.05       | 92.87        | 89.58               | 3.29       | 76.04      |
|           | HRank [21]| 76.15        | 75.33               | 0.8        | 92.86        | 92.46               | 0.3        | 73.44      |
|           | Ours     | 76.13        | 75.24               | 0.89       | 92.86        | 92.46               | 0.4        | 87.75      |

pruning methods including LFPC [36], LRF [37], MainDP [9], HRank [21], DMC [6] and SCP [8]. On ResNet32, our method improves the accuracy by 0.85% after pruning 55.54% FLOPs, and outperforms LFPC and FPGM [38] at similar FLOPs reduction rates. When 62.42% FLOPs are pruned, the accuracy improvement of CAP is still better than that of LRF and MainDP. On ResNet56, we included additional competitive methods for comparison. At different target FLOPs reduction rates, CAP prunes more FLOPs and simultaneously yields higher accuracy boost than other methods. For instance, compared with the best-performing existing method SRR-GR [39], our method has better accuracy improvement (0.42% vs 0.37%) and higher pruning rate (56.06% vs 53.80%). When 65.64% FLOPs are pruned, CAP improves the accuracy by 0.33%, and outperforms LFPC (0.28% accuracy improvement) and MainDP (0.06% accuracy loss) at similar FLOPs drop rates. For ResNet110, our method achieves 0.39% accuracy improvement with 53.18% FLOPs reduced, while TAS [40] results in 0.64% accuracy loss when pruning similar proportion of FLOPs. CAP also shows advantage in producing much compact subnetworks, it improves the accuracy by 0.25% after removing 68.81% FLOPs, and outperforms HRank by a large margin on model accuracy.

We then evaluated our method on CIFAR-100 dataset, and the results are given in Table 2. For Resnet32, CAP improves the accuracy by 0.98% with 51.57% FLOPs reduction rate, while other methods shows accuracy loss even at lower FLOPs pruning rates. For instance, the accuracy loss of SFP [8] is as high as 1.4% when 41.5% FLOPs are pruned. When pruning 56.06% FLOPs on ResNet56, CAP still improves accuracy by 0.59%, and outperforms other methods including LCCL [41], SFP, FPGM [38] and TAS [40].

4.3 Results on ImageNet

On ImageNet dataset, we evaluated the performance of CAP on ResNet50 network, and compared it with current advanced methods (Table 3). Performance metrics such as Top-1 and Top-5 accuracy are adopted for comprehensive comparison. CAP is able to generate more compact networks while better preserve the model accuracy. For instance, Top-1 accuracy only decreases by 0.8% and Top-5 accuracy only decreases by 0.3% with 73.44% FLOPs reduction. By comparison, OTO [17] results in 1.00% loss on Top-1 and 0.4% loss on Top-5 accuracy with only 64.50% FLOPs reduction. CAP also yields a highly compact architecture with 87.75% FLOPs reduction, which causes only 0.89% Top-1 accuracy drop. At a cost of similar accuracy loss, CHIP [22] only prunes 62.8% FLOPs. Similar to our method, Zhuang et al. [10] also employs a polarization regularizer on channel-wise scaling factors to find redundant filters, it prunes 70% FLOPs but results in 2% accuracy loss. The better performance of CAP when compared to Zhuang et al. [10] may benefit from our modeling of non-uniform contribution of samples to filter importance in CAP. Our method also outperforms recently published methods proposed in [43] and [44]. Taken together, these results suggest our proposed method scales well to complex dataset.
Table 4: Performance comparison between uniform and non-uniform weights of samples. The evaluations are conducted on CIFAR-10 dataset to compress ResNet32 and ResNet56 networks. \(\text{ACC}_\downarrow\) represents the percent of accuracy drop after pruning. \(\text{FLOPs}_\downarrow\) represents FLOPs reduction rate.

| Network   | Weights   | Baseline Acc(%) | \(\text{ACC}_\downarrow\) (%) | \(\text{FLOPs}_\downarrow\) (%) |
|-----------|-----------|-----------------|-------------------------------|---------------------------------|
| ResNet32  | non-uniform | 92.94           | -0.78                         | 62.42                           |
|           | uniform    |                 | -0.21                         | 60.95                           |
| ResNet56  | non-uniform | 93.25           | -0.33                         | 65.64                           |
|           | uniform    |                 | -0.15                         | 66.11                           |

Figure 3: Comparison of CE loss between baseline and pruned networks. For each image in test dataset, CE loss is calculated for the baseline and pruned networks, respectively. Linear regression is employed to analyze the relationship between the loss of pruned and baseline networks (the fitted line is marked by red).

4.4 Ablation Study

The effectiveness of non-uniform weights of samples. Unlike existing methods that often assume a uniform contribution of samples to filter importance, we employ non-uniform weights for samples when measuring the importance scores of filters. The weight of each sample is defined based on the CE loss to reflect the complexity of the sample. To verify the effectiveness of proposed non-uniform weights, we make a comparison between the pruning results with uniform and non-uniform weights, and the results on CIFAR-10 dataset are shown in Table 4. With sample complexity considered, CAP is able to significantly enhance the accuracy of pruned models. For instance, at \(\sim61\%\) FLOPs reduction rate, only 0.21\% accuracy improvement on ResNet32 is obtained without modeling sample complexity.

To investigate how modeling of instance complexity affects performance of the pruned network, we compared CE loss of the baseline and pruned networks for ResNet56, ResNet110 and ResNet50 (as shown in Figure 3). Linear relationship between loss of pruned network and loss of baseline network was fitted, and the results suggest our proposed instance complexity-aware pruning method significantly decreases CE loss especially for hard samples, thus better preserves the model accuracy. As we give higher weights to hard samples when determining the soft masks, the network pruning is encouraged to better maintain the prediction capability on complex instances.

Table 5: Comparison of pruning results on ResNet32 and ResNet56 networks with and without the regularizer on soft masks. \(\text{ACC}_\downarrow\) represents the percent of accuracy drop after pruning. \(\text{FLOPs}_\downarrow\) represents FLOPs reduction rate.

| Network   | Regularizer | Baseline Acc(%) | \(\text{ACC}_\downarrow\) (%) | \(\text{FLOPs}_\downarrow\) (%) |
|-----------|-------------|-----------------|-------------------------------|---------------------------------|
| ResNet32  | with \(R(m)\) | 92.94           | -0.78                         | 62.42                           |
|           | without \(R(m)\) |                 | -0.09                         | 61.92                           |
| ResNet56  | with \(R(m)\) | 93.25           | -0.33                         | 65.64                           |
|           | without \(R(m)\) |                 | -0.15                         | 66.11                           |
Figure 4: The polarization effect of the proposed regularizer on soft masks. Histograms of the soft masks are plotted at different iterations during the training process. With the training iterations increase, the soft masks are gradually separated into two parts with a large margin between them, and finally the masks of some filters locate at near 0 and the remaining locate at 1.

Figure 5: Analysis of similarity between the soft masks across different batches. 100 batches of inputs and 200 filters are randomly selected for evaluation, each row represents a batch and each column denotes a filter. Values of the soft mask derived from different batches were compared for each filter. ResNet32, ResNet56 and ResNet110 were analyzed on CIFAR-10 dataset. The effectiveness of the regularizer on important scores.

To verify the effectiveness of our proposed regularizer on the soft masks of filters, we compared the results with and without the polarization regularizer. The evaluation was conducted on CIFAR-10 dataset to compress ResNet32 and ResNet56 networks. If the regularizer is not applied, there is no obvious gap in the mask distribution, therefore we manually select the threshold to make sure the desired FLOPs reduction rate is reached. As shown in Table 5, the proposed polarization regularizer yields better performing subnetworks that has higher accuracy and less FLOPs. For instance, with ~62% FLOPs reduction rate on ResNet32, applying the regularizer enables 0.78% accuracy improvement, and only 0.09% accuracy boost is achieved without the regularizer. Similar results are observed on ResNet56. The polarization effect of the proposed regularizer is shown in Figure 4, the filters are gradually separated into two parts when the training iterations increases, and finally a large margin between the two sets of filters is observed. These results demonstrate our proposed regularizer is effective in identifying important filters, and helpful to find a sweet spot that clearly separate the filters into two parts.

As our method uses the weighted average of soft masks calculated for each batch of inputs to scale channel-wise outputs of the pruned network, and different batches may end up with differently distributed soft masks, we further evaluated the similarity between the soft masks of different batches after the model converges. Specifically, the evaluation was performed on CIFAR-10 dataset, and we randomly selected 100 batches and 200 filters for comparison. The results on Figure 5 show different batches yield highly consistent distribution of the soft masks, and the filters are consistently separated into two parts with a large margin across different batches, which suggests our method has high robustness in reasoning the redundant filers and only one batch of inputs is required to prune the filters after model converges.

5 Conclusions

In this paper, we propose an instance complexity-aware channel pruning method called CAP to compress a large baseline network. The main novelty of CAP lies in its unique feature of modeling non-uniform contribution of different samples to filter importance. By introducing instance-complexity
related weights, the importance scores of filters are defined as the weighted sum of sample-specific soft masks, thus hard samples are given higher weights in determining which filters are important for classification. In addition, a new regularizer is proposed in CAP to push the important scores towards either 0 or 1, such that a sweet spot can be easily found to divide the filters into important and redundant filters. Performance evaluations on various network architectures and datasets demonstrate CAP has advantages over the SOTAs in preserving model accuracy and pruning FLOPs.

One of the limitations of CAP lies in the fact that it still requires a pretrained baseline network to calculate the weights of instances based on CE loss, and pretraining large networks requires extensive computations. There may be other strategies to measure the complexity of instances, and we plan to investigate this in future works. In addition, we currently test our method on some common datasets, and we plan to apply our method to more datasets to examine its effectiveness on other research domains.

References

[1] Chao Dong, Chen Change Loy, Kaiming He, and Xiaou Tang. Image super-resolution using deep convolutional networks. *IEEE transactions on pattern analysis and machine intelligence*, 38(2):295–307, 2015.

[2] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 580–587, 2014.

[3] Shaohui Lin, Rongrong Ji, Yuchao Li, Yongjian Wu, Feiyue Huang, and Baochang Zhang. Accelerating convolutional networks via global & dynamic filter pruning. In *IJCAI*, volume 2, page 8, 2018.

[4] 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.

[5] 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*, volume 32. Curran Associates, Inc., 2019.

[6] Shangqian Gao, Feihu Huang, Jian Pei, and Heng Huang. Discrete model compression with resource constraint for deep neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1899–1908, 2020.

[7] Jaedeok Kim, Chiyoun Park, Hyun-Joo Jung, and Yoonsuck Choe. Plug-in, trainable gate for streamlining arbitrary neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 4452–4459, 2020.

[8] Minsoo Kang and Bohyung Han. Operation-aware soft channel pruning using differentiable masks. In *International Conference on Machine Learning*, pages 5122–5131. PMLR, 2020.

[9] Yehui Tang, Yunhe Wang, Yixing Xu, Yiping Deng, Chao Xu, Dacheng Tao, and Chang Xu. Manifold regularized dynamic network pruning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5018–5028, 2021.

[10] Tao Zhuang, Zhiyuan Zhang, Yuheng Huang, Xiaoyi Zeng, Kai Shuang, and Xiang Li. Neuron-level structured pruning using polarization regularizer. In *NeurIPS*, 2020.

[11] Xin Dong, Shangyu Chen, and Sinno Jialin Pan. Learning to prune deep neural networks via layer-wise optimal brain surgeon. *arXiv preprint arXiv:1705.07565*, 2017.

[12] Song Han, Jeff Pool, John Tran, and William J Dally. Learning both weights and connections for efficient neural networks. *arXiv preprint arXiv:1506.02666*, 2015.

[13] Daiki Chijiwa, Shin’ya Yamaguchi, Yasutoshi Ida, Kenji Umakoshi, and Tomohiro INOUE. Pruning randomly initialized neural networks with iterative randomization. In A. Beygelzimer, Y. Dauphin, P. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, 2021.

[14] 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.
[15] Huidong Liu, Fang Du, Xiaofen Tang, Hao Liu, and Zhenhua Yu. Network architecture reasoning via deep deterministic policy gradient. In 2020 IEEE International Conference on Multimedia and Expo (ICME), pages 1–6, 2020.

[16] Zhongzhan Huang, Wenqi Shao, Xinjiang Wang, Liang Lin, and Ping Luo. Rethinking the pruning criteria for convolutional neural network. In A. Beygelzimer, Y. Dauphin, P. Liang, and J. Wortman Vaughan, editors, Advances in Neural Information Processing Systems, 2021.

[17] Tianyi Chen, Bo Ji, Tianyu DING, Biyi Fang, Guanyi Wang, Zihui Zhu, Luming Liang, Yixin Shi, Sheng Yi, and Xiao Tu. Only train once: A one-shot neural network training and pruning framework. In A. Beygelzimer, Y. Dauphin, P. Liang, and J. Wortman Vaughan, editors, Advances in Neural Information Processing Systems, 2021.

[18] Shengcao Cao, Xiaofang Wang, and Kris M. Kitani. Learnable embedding space for efficient neural architecture compression. In International Conference on Learning Representations, 2019.

[19] Lucas Liebenwein, Cenk Baykal, Harry Lang, Dan Feldman, and Daniela Rus. Provable filter pruning for efficient neural networks. arXiv preprint arXiv:1911.07412, 2019.

[20] Ting-Wu Chin, Ruizhou Ding, Cha Zhang, and Diana Marculescu. Towards efficient model compression via learned global ranking. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 1518–1528, 2020.

[21] 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.

[22] Yang Sui, Miao Yin, Yi Xie, Huy Phan, Saman Aliari Zonouz, and Bo Yuan. CHIP: CHannel independence-based pruning for compact neural networks. In A. Beygelzimer, Y. Dauphin, P. Liang, and J. Wortman Vaughan, editors, Advances in Neural Information Processing Systems, 2021.

[23] 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/CVF Conference on Computer Vision and Pattern Recognition, pages 2790–2799, 2019.

[24] Shaopeng Guo, Yujie Wang, Quanquan Li, and Junjie Yan. Dmcp: Differentiable markov channel pruning for neural networks. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 1539–1547, 2020.

[25] Xiaofeng Ruan, Yufan Liu, Bing Li, Chunfeng Yuan, and Weiming Hu. Dpfps: Dynamic and progressive filter pruning for compressing convolutional neural networks from scratch. In Proceedings of the AAAI Conference on Artificial Intelligence, volume 35, pages 2495–2503, 2021.

[26] Zhao Zhong, Junjie Yan, Wei Wu, Jing Shao, and Cheng-Lin Liu. Practical block-wise neural network architecture generation. In Proceedings of the IEEE conference on computer vision and pattern recognition, pages 2423–2432, 2018.

[27] Hieu Pham, Melody Guan, Barret Zoph, Quoc Le, and Jeff Dean. Efficient neural architecture search via parameters sharing. In International conference on machine learning, pages 4095–4104. PMLR, 2018.

[28] Esteban Real, Sherry Moore, Andrew Selle, Saurabh Saxena, Yutaka Leon Suematsu, Jie Tan, Quoc V Le, and Alexey Kurakin. Large-scale evolution of image classifiers. In International Conference on Machine Learning, pages 2902–2911. PMLR, 2017.

[29] Hanxiao Liu, Karen Simonyan, Oriol Vinyals, Chrisantha Fernando, and Koray Kavukcuoglu. Hierarchical representations for efficient architecture search. arXiv preprint arXiv:1711.00436, 2017.

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

[31] Ji Lin, Yongming Rao, Jiwen Lu, and Jie Zhou. Runtime neural pruning. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, Advances in Neural Information Processing Systems, volume 30. Curran Associates, Inc., 2017.
[32] 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 (ICCV), Oct 2017.

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

[34] 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.

[35] 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.

[36] 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.

[37] Donggyu Joo, Eojindl Yi, Sunghyun Baek, and Junmo Kim. Linearly replaceable filters for deep network channel pruning. In Proceedings of the AAAI Conference on Artificial Intelligence, volume 35, pages 8021–8029, 2021.

[38] Y. He, P. Liu, Z. Wang, Zhilan Hu, and Y. Yang. Filter pruning via geometric median for deep convolutional neural networks acceleration. 2018.

[39] Zi Wang, Chengcheng Li, and Xiangyang Wang. Convolutional neural network pruning with structural redundancy reduction. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 14913–14922, 2021.

[40] Xuanyi Dong and Yi Yang. Network pruning via transformable architecture search. arXiv preprint arXiv:1905.09717, 2019.

[41] Xuanyi Dong, Junshi Huang, Yi Yang, and Shuicheng Yan. More is less: A more complicated network with less inference complexity. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 5840–5848, 2017.

[42] Wenxiao Wang, Minghao Chen, Shuai Zhao, Long Chen, Jinming Hu, Haifeng Liu, Deng Cai, Xiaofei He, and Wei Liu. Accelerate cnns from three dimensions: A comprehensive pruning framework. In International Conference on Machine Learning, pages 10717–10726. PMLR, 2021.

[43] Xiaohan Ding, Tianxiang Hao, Jianchao Tan, Ji Liu, Jungong Han, Yuchen Guo, and Guiguang Ding. Resrep: Lossless cnn pruning via decoupling remembering and forgetting. In Proceedings of the IEEE/CVF International Conference on Computer Vision, pages 4510–4520, 2021.

[44] Liyang Liu, Shilong Zhang, Zhanghui Kuang, Aojun Zhou, Jing-Hao Xue, Xinjiang Wang, Yimin Chen, Wenming Yang, Qingmin Liao, and Wayne Zhang. Group fisher pruning for practical network compression. In International Conference on Machine Learning, pages 7021–7032. PMLR, 2021.