Entropy Induced Pruning Framework for Convolutional Neural Networks

Yiheng Lu, Ziyu Guan, Yaming Yang, Wei Zhao*, Maoguo Gong, Cai Xu

Key Laboratory of Collaborative Intelligence Systems, Ministry of Education,
Xidian University, Xi'an, China
lyhxdu@gmail.com, zyguan@xidian.edu.cn, yym@xidian.edu.cn, ywzhao@mail.xidian.edu.cn, gong@ieee.org,
cxu@xidian.edu.cn

Abstract

Structured pruning techniques have achieved great compression performance on convolutional neural networks for image classification tasks. However, the majority of existing methods are sensitive with respect to the model parameters, and their pruning results may be unsatisfactory when the original model is trained poorly. That is, they need the original model to be fully trained, to obtain useful weight information. This is time-consuming, and makes the effectiveness of the pruning results dependent on the degree of model optimization. To address the above issue, we propose a novel metric named Average Filter Information Entropy (AFIE). It decomposes the weight matrix of each layer into a low-rank space, and quantifies the filter importance based on the distribution of the normalized eigenvalues. Intuitively, the eigenvalues capture the covariance among filters, and therefore could be a good guide for pruning. Since the distribution of eigenvalues is robust to the updating of parameters, AFIE can yield a stable evaluation for the importance of each filter no matter whether the original model is trained fully. We implement our AFIE-based pruning method for three popular CNN models of AlexNet, VGG-16, and ResNet-50, and test them on three widely-used image datasets MNIST, CIFAR-10, and ImageNet, respectively. The experimental results are encouraging. We surprisingly observe that for our methods, even when the original model is trained with only one epoch, the AFIE score of each filter keeps identical to the results when the model is fully-trained. This fully indicates the effectiveness of the proposed pruning method.

Introduction

Convolutional Neural Networks (CNNs) have achieved impressive success on image classification with large model size (Krizhevsky, Sutskever, and Hinton 2012; Szegedy et al. 2015; Simonyan and Zissermann 2015; He et al. 2016). However, their growing scale imposes huge pressure on computational resources, and may easily lead to the overfitting issue due to the redundancy across the whole network. Therefore, network pruning techniques are studied comprehensively by researchers, to cut off redundant computation branches from original models.

Generally, there are two categories of pruning strategies, i.e., structured pruning (He, Zhang, and Sun 2017; Luo, Wu, and Lin 2017; Molchanov et al. 2017; Liu et al. 2017) and unstructured pruning (LeCun, Denker, and Solla 1990; Hassibi et al. 1993; Jonathan and Michael 2019). Structured pruning is to remove the whole filters from each model layer. Unstructured pruning is to prune partial parameters from the original model. Structured pruning has been proved more efficient than unstructured pruning because structured pruning can eliminate the number of feature maps, which can reduce the consumption of FLOPs apparently (Liu et al. 2019b). Therefore, in this work, we mainly focus on structured pruning methods.

Structured pruning methods can be further classified into Layers-Importance-Supported (LIS) methods and Filters-Importance-Supported (FIS) methods. LIS methods aim to evaluate the layer importance at first, and then remove filters according to the layer importance. For example, (Chin, Zhang, and Marculescu 2018; Suau et al. 2018; Li et al. 2017) utilize different weight-oriented strategies to evaluate the importance of each convolutional layer based on fully-trained models. Differently, FIS methods focus on evaluating the importance of each filter straightly. For example, (Molchanov et al. 2017; Liu et al. 2017; Ye et al. 2018) evaluate the importance of each filter by the gradient information and BatchNorm neurons, which can achieve global pruning automatically.

However, both LIS methods and FIS methods above need the original model to be fully trained, to obtain enough meaningful parameters. This makes the pruning operation time-consuming and resource-inefficient. Moreover, these pruning methods' effectiveness depends on the degree of the model optimization. They may easily yield degenerated pruning results if the original model is not fully trained.

To address the above issue, in this paper, we propose the metric Average Filter Information Entropy (AFIE), which can compute a reliable evaluation score for each filter even when the original model is trained with only several epochs. In practice, AFIE can even yield the same evaluation of filters for a fully-trained model and a one-epoch-trained model.

As shown in Figure 1, the overall procedure consists of three steps. First, we decompose the weight matrix $\widetilde{\mathbf{M}}$ of a specified convolutional layer into a low-rank space. Then,

*Corresponding Author

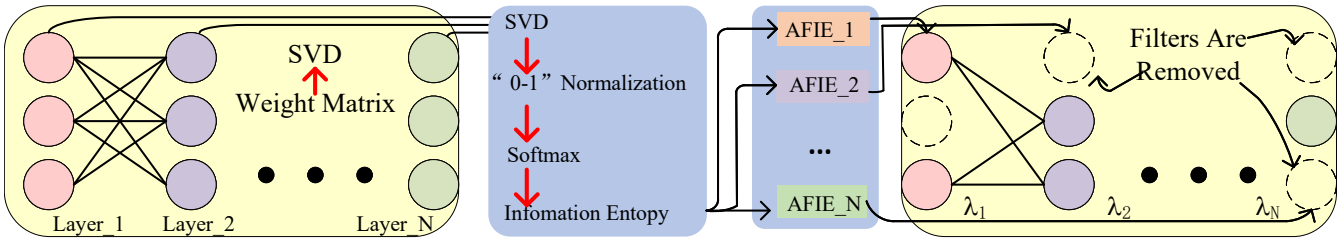

Figure 1: The calculation of AFIE for each convolutional layer. The weight matrix of each convolutional layer is decomposed by SVD, and then yields an eigenvalue matrix. Simultaneously, all eigenvalues will be normalized by "0-1" scaling and softmax to be projected into a distribution. Finally, we use the information entropy to quantify the distribution of the normalized eigenvalues, obtaining AFIE. Based on the AFIE scores, we are able to allocate the corresponding pruning ratios $\lambda_1, \lambda_2, ..., \lambda_N$ for each of the convolutional layers.

we normalize the eigenvalues into a distribution by the "0-1" normalization and softmax function. Finally, we adopt the information entropy to quantify the eigenvalues' distribution, obtaining the AFIE score. The pruning ratio for each layer is assigned according to its AFIE score.

Intuitively, in our method, the information entropy can describe the distribution of the eigenvalues. A low information entropy means that the distribution is relatively sharp, i.e., several top eigenvalues account for most of the intensity. This indicates that the rank of the weight matrix $\widetilde{\mathbf{M}}$ is low. The lower the rank of $\widetilde{\mathbf{M}}$, the higher the redundancy of the parametric linear transformation (i.e., convolutional filters within the current layer). For an extreme case, when all the filters are linearly correlated, there will be only one positive eigenvalue, and the others are 0's. It means that the rank of $\widetilde{\mathbf{M}}$ is 1. Obviously, in this case, we should keep only one filter but remove all the others. On the contrary, when $\widetilde{\mathbf{M}}$ are full rank, we tend to keep all the filters.

We theoretically analyze that the updating of parameters between the one-epoch-trained and fully-trained models is Lipschitz continuous with the gradient of the one-epoch-trained model. It means that we can get a similar distribution of parameters for a fully-trained model and a one-epoch-trained model, when the gradient of the one-epoch model is small. This facilitates our AFIE-based method to yield a stable evaluation.

We conduct extensive experiments to verify the effectiveness of the proposed AFIE. Specifically, we apply our AFIE-based pruning framework to popular CNN models AlexNet, VGG-16, and ResNet-50, and test them on MNIST, CIFAR-10, and ImageNet. The experimental results show that our pruned model is able to recover the accuracy of the original model, and achieve competitive results in terms of reducing parameters and FLOPs.

In summary, we make three major contributions in this work.

- We observe that previous weight-oriented methods are sensitive to the update of parameters, which can not yield convincing pruning results when the original model is not trained well.
- We propose an entropy-based method to describe the importance of each filter without considering the weight information. Consequently, we can obtain an identical importance evaluation of each filter no matter whether the original model is fully-trained.
- We conduct extensive experiments. It turns out that for our framework, even when the original model is trained with only one epoch, the pruning results are comparable with that of previous methods.

Related Work

In this section, we briefly introduce the existing Layer-Importance-Supported (LIS) pruning methods and the existing Filters-Importance-Supported (FIS) pruning methods.

LIS Pruning Methods

Study (Ding et al. 2021) utilizes a long short-term memory (LSTM) to learn the hierarchical characteristics of a network and generate a global network pruning scheme. Study (Li et al. 2017) explores the sensitiveness of each convolutional layer through several layer-wise pruning experiments with l_1-norm. This work stresses the importance of sensitive layers and removes filters from insensitive layers. Similarly, study (Mao et al. 2017) proposes a coarse-grained pruning method to prune the layers iteratively through sensitivity analysis. Study (Jiang et al. 2018) achieves layer pruning by minimizing the reconstruction error of nonlinear units. It employs a greedy algorithm to remove trivial neurons. Study (Chin, Zhang, and Marculescu 2018) treats the pruning problems as a unified problem with layer scheduling and ranking problem. It compensates for the approximation error across layers during derivation with the assistance of meta-learning. Study (Suau et al. 2018) utilizes principal filter analysis to yield a compact model by exploiting the intrinsic correlation of filter responses within layers. Study (Yu et al. 2018) proposes a final layer response mechanism to propagate layer importance from the last layer to the shallow layer by minimizing the reconstruction error. All the above-mentioned LIS methods evaluate the importance of layers and filters, and they have achieved impressive pruning results on the image classification task. However, LIS methods usually require a well-pre-trained model to provide useful information for supporting the evaluation.

FIS Pruning Methods

Study (Liu et al. 2022) proposes a simple-yet-effective method called discrimination-aware channel pruning (DCP) to choose the channels that actually contribute to the discriminative power. Study (He, Zhang, and Sun 2017) introduces a two-phase channel pruning technique with LASSO regression, which exploits the redundancy inter feature maps. Study (Luo, Wu, and Lin 2017) proposes the ThiNet to compress models at both training and inference processes at the channel level. Study (He et al. 2018) uses the soft filter technique to remove filters with large model capacity, which can learn more useful information from the training data, as well as reduce the dependence on the pre-trained model. Study (Ye et al. 2018) forces the output of a part of the filters as a constant. Then, the constant is removed by adjusting the bias of impacting layers. Study (Molchanov et al. 2017) simplifies the pruning process through Taylor expansion with the first order. Then the importance of each filter is evaluated by the products of the weight value and the gradient. Study (Liu et al. 2017) regularizes BatchNorm layers, and corresponding filters would be removed if BatchNorm neurons were evaluated with small value. Study (Liu et al. 2019a) jointly utilizes meta-learning and evolutionary algorithm to prune a well-trained model automatically. Additionally, some sparsity constraints on filters are also explored comprehensively. Study (Zhou, Alvarez, and Porikli 2016) and study (Alvarez and Salzmann 2016) add extra sparsity on a group of filters, which can remove less important filters by iterative pruning process. Study (Huang and Wang 2018) introduces a scale factor with sparsity constraints to select some useful filters within the original architecture, which avoids the heavy searching process compared with previous structure sparsity methods. However, these methods above also require a well-pre-trained model to clarify the status of each filter, and the evaluation may be twisted with the updating of parameters from the original model.

Methodology

The previous work (Huang et al. 2021) concludes that the parameters within each filter tend to show linearity and the parameters among the filters tend to show non-linearity with the increase of the training epochs. Therefore, we can regard each filter as one base dimension of the parameter space, as well as project the parameter space of each layer into the low-rank space along all the dimensions. The eigenvalues within the low-rank space can reflect the property of the original parameter space, and consequently, the information entropy of the parameter space can be quantified by the distribution of eigenvalues. The redundancy of the filters within the current convolutional layer can be measured by the information entropy due to the equivalent linear transformation between the parameter space and low-rank space.

The overall process of AFIE is illustrated in Figure 1. The pruning procedure consists of three main steps, i.e., decomposition, normalization, and information entropy calculation. Motivated by the work from (Zhang et al. 2016), the weight matrix can be decomposed by SVD. Then, the obtained eigenvalues are normalized by softmax to map each eigenvalue into possibilities. Finally, the information entropy is calculated based on the obtained possibilities. We can use the obtained information entropy to quantify the filter importance. Intuitively, if the distribution of normalized eigenvalues is flat, then, less redundancy can be found because the projection of the original model has similar strength along each dimension.

We also analyze the efficacy of AFIE mathematically, as well as explore the updating of the weight matrix by visualizing the variation of gradients and parameters (See Appendix). It implies good consistency of AFIE for evaluating the importance of each filter between the fully-trained model and the one-epoch-trained model, i.e., we can obtain a stable evaluation of AFIE no matter whether the original model is fully trained.

Low-Rank Space Projection

Usually, the parameters within a convolutional layer can be properly packed as a four-dimensional weight tensor $\widetilde{\mathbf{M}}^{(\mathcal{I}\times\mathcal{O}\times\mathcal{H}\times\mathcal{W})}$, where $\mathcal{I}$ and $\mathcal{O}$ are the number of input and output filters, and $\mathcal{H}$ and $\mathcal{W}$ are the height and width of the filters. Please refer to the illustration in Figure 1. It is difficult to straightly decompose $\widetilde{\mathbf{M}}$. Therefore, we first fold $\widetilde{\mathbf{M}}$ as a two-dimensional matrix $\mathbf{M}$ by averaging the dimensions $\mathcal{H}\times\mathcal{W}$, denoted as follows:

$$\mathbf{M}_l^{(\mathcal{I}^*\times\mathcal{O}^*)} = Aver_{(\mathcal{H},\mathcal{W})}(\widetilde{\mathbf{M}}_l^{(\mathcal{I}\times\mathcal{O}\times\mathcal{H}\times\mathcal{W})}), \tag{1}$$

Usually, SVD can be used for principal components analysis, which yields an eigenvalue matrix to represent the property of the original matrix. Here, we adopt SVD to perform the low-rank projection for $\mathbf{M}_l^{(\mathcal{I}^*\times\mathcal{O}^*)}$:

$$(\mathbf{U}*\mathbf{s}*\mathbf{V})_{(l)} = SVD(\mathbf{M}_l^{\mathcal{I}^*\times\mathcal{O}^*}). \tag{2}$$

where $\mathbf{s}$ stands for the decomposed eigenvalues, which is a diagonal matrix that has sorted eigenvalues along the diagonal. $\mathbf{U}$ and $\mathbf{V}$ are the left singular matrix and right singular matrix, respectively.

Originally, the eigenvalues can be used to describe the zoom level of each axis for the standard orthogonal basis. Here, we aim to use the distribution of eigenvalues to represent the importance of the layers because the eigenvalues maintain the majority of information of the original matrix. Creatively, we redefine the eigenvalues as the projection of useful information for the specified layer at different dimensions. The number and absolute value of eigenvalues stand for the dimensions and strength of the projection, respectively. Then we can obtain efficient information about the specified layer by analyzing the decomposed eigenvalues.

Normalization of the Eigenvalues

In order to perform the comparison between each convolutional layer at the same magnitude, we apply "0-1" normalization to the eigenvalues, so as to scale each value to the range [0,1]. This process is described as follows:

$$s_l^{(i,norm)} = \frac{s_l^i - s_l^{min}}{s_l^{max} - s_l^{min}}, \quad s.t. \quad i = 1, 2, ..., p_l. \tag{3}$$

where $s_l^{(i,norm)}$ and s_l^i stand for i-th normalized and original eigenvalues for layer l, respectively, s_l^{min} and s_l^{max} stand for the minimum and maximum value within $\mathbf{s}_l$, respectively, and p_l is the number of eigenvalues for layer l.

To constraint the summation of all the eigenvalues to 1, we further perform softmax normalization on these eigenvalues, as follows:

$$s_l^{(i,soft)} = \frac{\exp^{s_l^{(i,norm)}}}{\sum_{i=1}^{p_l} \exp^{s_l^{(i,norm)}}}. \tag{4}$$

Thus, we can make the parameter distributions of different convolutional layers comparable in the low-rank space. This facilitates the quantification of the layer importance, as described in the next step.

Average Filter Information Entropy

We borrow the concept of "energy" from the Boltzmann machine to transfer each eigenvalue into possibilities, which can describe how possible a projection along one dimension will respond. Higher energy means the specified layer contains more useful information along one projected dimension. We can quantify the energy for a specified layer by the information entropy, which can disentangle the importance of each layer in a mathematical way. The computation of information entropy is described as follows:

$$H(x) = -\sum_{x \in \Psi} p(x) \log p(x), \tag{5}$$

where x and $p(x)$ stand for a specified status and the corresponding probability, Ψ is the set of all possible statuses within a fixed system.

(Zhang et al. 2016) explores the low-rank decomposition for each convolutional layer to speed up the training of the network, which motivates us to adopt the decomposed eigenvalues to represent all the possible statuses (projection) for describing useful information. We use the information entropy to measure the distribution of the normalized eigenvalues for layer l, as follows:

$$K_l = -\sum_{i=1}^{p_l} s_l^{(i,soft)} \log s_l^{(i,soft)}. \tag{6}$$

Here, we can use K_l to describe the importance of layer l. As we have explained previously, a lower value K_l means that the distribution of eigenvalues is sharp. It implies that the rank of the weight matrix $\mathbf{M}$ is low, indicating that there is redundant information in the parametric linear transformation. To reduce this redundancy, we need to remove some parameters, i.e., some filters.

Obviously, in the above Equation (6), both $s_l^{(i,soft)}$ and p_l rig the calculation of K_l, which means the number of filters will impose huge effect to evaluate the total useful information for a layer. Inspired by a previous work (Liu et al. 2019b) which concludes that the representation ability of CNNs is dominated by the structure of the model, here, we reasonably assume that *each filter has the same importance within one layer* because all the filters within a convolutional layer usually have the same structure. Thus, we can obtain the importance of each filter for layer l by dividing the K_l over the number of filters c_l, arriving at the proposed AFIE metric for layer l:

$$AFIE_l = \frac{K_l}{c_l}. \tag{7}$$

The obtained $AFIE_l$ describes the importance of each filter for layer l, and thus the importance of all the filters across all layers can be quantified.

Filters Pruning with AFIE

After obtaining the AFIE scores of all the layers, we allocate the pruning ratio for each layer according to the following equation:

$$\begin{aligned} &\lambda_l = \lambda_{min} \frac{AFIE_{max}}{AFIE_l}, \\ s.t. \quad &\sum_{l=1}^{l=N} \lambda_l p_l = \lambda^* p^*, \end{aligned} \tag{8}$$

where λ_l stands for the pruning ratio for layer l, λ^* and p^* respectively stand for the specified overall pruning ratio and the total number of filters for the original model, and N is the number of the convolutional layers. Thus, we can calculate the pruning ratio of each layer by Equation (8). Notably, the layer with the maximum AFIE, i.e. $AFIE_{max}$ should be assigned the minimum pruning ratio, i.e. λ_{min}. According to Equation (8), we have the following derivation:

$$\begin{aligned} \lambda^* p^* &= \sum_{l}^{N} \lambda_{min} \frac{AFIE_{max}}{AFIE_l} p_l, \\ &\Downarrow \\ \lambda_{min} &= \sum_{l}^{N} \frac{\lambda^* p^* AFIE_l}{AFIE_{max} p_l}, \end{aligned} \tag{9}$$

In particular, we reserve a small set of filters (1% in this work) within each convolutional layer to maintain the integrity of the topology for the original model. Otherwise, the gradient information can not flow from deep layers to shallow layers when a whole layer is removed. Thus, we regulate Equation (9) as:

$$\lambda_l = \begin{cases} \lambda_{min} \frac{AFIE_{max}}{AFIE_l}, & \lambda_{min} \frac{AFIE_{max}}{AFIE_l} < 1, \\ 0.99, & \lambda_{min} \frac{AFIE_{max}}{AFIE_l} \geq 1. \end{cases} \tag{10}$$

We can determine the pruning ratio of each convolutional layer quickly by solving the above equations. This process avoids massive iterative pruning and retraining, saving many computational resources.

Experiments

Implementation of Filters Removal

After obtaining the AFIE scores of all the layers, we randomly remove filters within each layer, under the guidance

Model	Epochs	Top1-Acc	AFIE			
			$AFIE_1$	$AFIE_2$	$AFIE_3$	$AFIE_{4-5}$
AlexNet	1	0.955	0.015	0.022	0.014	0.022
	10	0.988	0.015	0.022	0.014	0.022
	20	0.992	0.015	0.022	0.014	0.022

Table 1: The evaluation of each convolutional layer for AlexNet on MNIST.

Model	Epochs	Top1-Acc	AFIE							
			$AFIE_1$	$AFIE_2$	$AFIE_3$	$AFIE_4$	$AFIE_5$	$AFIE_6$	$AFIE_7$	$AFIE_{8-13}$
VGG-16	1	0.435	0.016	0.064	0.032	0.038	0.019	0.022	0.011	0.012
	50	0.905	0.016	0.064	0.032	0.038	0.019	0.022	0.011	0.012
	150	0.9335	0.016	0.064	0.032	0.038	0.019	0.022	0.011	0.012

Table 2: The evaluation of each convolutional layer for VGG-16 on CIFAR-10.

Model	Epochs	Top1-Acc	AFIE			
			$AFIE_{1-3}$	$AFIE_{4-7}$	$AFIE_{8-13}$	$AFIE_{14-16}$
ResNet-50	1	0.435	0.064	0.038	0.022	0.012
	50	0.646	0.064	0.038	0.022	0.012
	150	0.758	0.064	0.038	0.022	0.012

Table 3: The evaluation of each convolutional layer for ResNet-50 on ImageNet.

Model	Epochs	Par-O	FLOPs-O	Pru-R	Par-P	FLOPs-P	Top1-Acc
AlexNet	1	3.52M	1.3×10^7	0.7	0.83M	1.0×10^7	0.9902
	20				0.83M	1.0×10^7	0.9920
VGG-16	1	33.65M	3.3×10^8	0.65	19.15M	1.5×10^8	0.9312
	150				19.15M	1.5×10^8	0.9335
ResNet-50	1	25.56M	4.12×10^9	0.3	21.88M	3.70×10^9	0.745
	150				21.88M	3.70×10^9	0.758

Table 4: The performance of slimmed models for AlexNet, VGG-16, and ResNet-50. The notations "Par-O", "FLOPs-O" "Par-P", and "FLOPs-P" stand for the parameters, and FLOPs for the original model and the pruned model, respectively.

Epochs	Model	Methods	Par-O	FLOPs-O	Pru-R	Par-P	FLOPs-P	Top1-Acc
1	AlexNet	ThiNet (Luo, Wu, and Lin 2017)	3.52M	1.3×10^7	0.7	1.45M	1.0×10^7	0.9827
		DCP (Liu et al. 2022)				1.22M	1.0×10^7	0.9854
		SEP (Ding et al. 2021)				1.20M	1.0×10^7	0.9891
		l_1 (Li et al. 2017)				1.38M	1.0×10^7	0.9865
		Network Slimming (Liu et al. 2017)				1.31M	1.0×10^7	0.9863
		Taylor (Molchanov et al. 2017)				0.75M	1.0×10^7	0.9886
		AFIE				**0.83M**	$\mathbf{1.0\times10^7}$	**0.9902**
	VGG-16	ThiNet (Luo, Wu, and Lin 2017)	33.65M	3.3×10^8	0.65	20.17M	1.7×10^8	0.9179
		DCP (Liu et al. 2022)				20.21M	1.6×10^7	0.9216
		SEP (Ding et al. 2021)				19.96M	1.0×10^7	0.9293
		l_1 (Li et al. 2017)				21.64M	1.8×10^8	0.9155
		Network Slimming (Liu et al. 2017)				19.95M	1.7×10^8	0.9237
		Taylor (Molchanov et al. 2017)				19.15M	1.1×10^8	0.9023
		AFIE				**19.15M**	$\mathbf{1.5\times10^8}$	**0.9312**
	ResNet-50	ThiNet (Luo, Wu, and Lin 2017)	25.56M	4.12×10^9	0.3	23.23M	3.81×10^9	0.735
		DCP (Liu et al. 2022)				22.69M	3.76×10^9	0.743
		SEP (Ding et al. 2021)				21.98M	3.58×10^9	0.739
		l_1(Li et al. 2017)				23.44M	3.85×10^9	0.722
		Network Slimming (Liu et al. 2017)				22.94M	3.78×10^9	0.736
		Taylor (Molchanov et al. 2017)				21.16M	3.47×10^9	0.719
		AFIE				**21.88M**	$\mathbf{3.70\times10^9}$	**0.745**

Table 5: The comparison of different pruning methods for AlexNet, VGG-16, and ResNet-50 on MNIST, CIFAR-10, and ImageNet, respectively.The 1 epoch means original models are trained poorly.

of these AFIE scores (See Appendix). There are two categories of pruning strategies for removing filters, as follows.

The iterative pruning strategy removes a part of the filters in several iterations. This requires a dynamic evaluation of AFIE. The whole pruning process continues until the overall pruning ratio reaches the specified λ^*. However, iterative pruning may lead the pruned model to be trapped in the local optima, since AFIE scores need to be re-calculated during each iteration.

The one-shot pruning strategy removes all the filters at once from all layers. Therefore, AFIE scores only need to be calculated once. Obviously, one-shot pruning avoids the re-calculation of AFIE for each convolutional layer, which can improve the efficiency of filter removal, as well as skip the trap of local optima. Therefore, in the experiments, we adopt one-shot pruning.

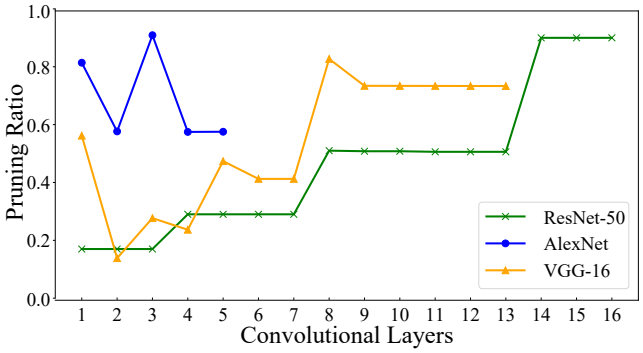

Figure 2: The pruning ratio of each convolutional layer for AlexNet, VGG-16, and ResNet-50 on MNIST, CIFAR-10, and ImageNet when the λ^* is set as 0.7, 0.65, and 0.3, respectively.

Effectiveness Analysis of AFIE

We implement our pruning framework AFIE based on AlexNet, VGG-16, and ResNet-50, and test them on MNIST, CIFAR-10, and ImageNet. First, we show the AFIE scores of each convolutional layer, for models that are trained with different epochs. The results are shown in Table 1, Table 2 and Table 3. As we can see, the AFIE stays unchanged no matter how many epochs the original model is trained. Even when the original model is trained for only one epoch, our method can still learn effective AFIE scores. Therefore, we can use the proposed AFIE to evaluate the importance of each filter without considering the interference of weight information. Additionally, we conduct an ablation study about the effect of training epochs of original models to show the effectiveness of AFIE for pruning tasks.

Then, we compare the performance with several previous structured pruning methods. The result is shown in Table 5 and Table 6. As we can see, our AFIE method can recover the accuracy of baselines on all the datasets. Particularly, AFIE is able to reduce the parameters and FLOPs than all the baselines except for the Taylor baseline. Nevertheless, Taylor typically sacrifices more accuracy. Therefore, our AFIE is a competitive method for network pruning.

Additionally, we test the performance of slimmed models when the original model is trained with different epochs. The experiments show that there are merely tiny sacrifices of accuracy for AlexNet, VGG-16, and ResNet-50 on MNIST, CIFAR-10, and ImageNet, respectively, as well as keeping the same reductions on parameters and FLOPs.

Specifically, we set the training epochs as 1 and 150 for VGG-16 and ResNet-50, as well as 1 and 20 for AlexNet. After pruning with AFIE, all models will be trained fully to recover the accuracy. The results are shown in Table 4. Obviously, we observe that all slimmed models can maintain the same reductions in parameters and FLOPs because AFIE can yield the same evaluation of importance for each layer, and then slimmed models can keep the same architecture no matter whether the original model is fully trained. With respect to the accuracy, it can recover to baseline when the original model is fully trained, and the sacrifice of accuracy is less than 1% when the original model is only trained with 1 epoch. Therefore, AFIE is a novel metric to prune CNNs at the early training stage to save massive computation resources.

Overall Comparison Against Baselines

Filters Pruning for AlexNet on MNIST AlexNet is constructed by 5 convolutional layers with BachNorm embedded. As shown in Table 1, the original model can achieve 99.20% top1-accuracy with 20 training epochs on MNIST. We set the overall pruning ratio λ^* to 70%, and specify the specific pruning ratio for each layer according to the Equation (8). To maximally demonstrate the good consistency of AFIE, we prune AlexNet when it is trained with only 1 epoch and 20 epochs, respectively. The results in Table 4 show that AFIE can achieve the same reductions of parameters and FLOPs for AlexNet when the overall pruning ratio is specified. Simultaneously, The accuracy only drops merely when the original model is trained with one epoch.

Figure 2 illustrates the details of filter pruning. As shown, Conv2 is assigned with the highest pruning ratio, i.e., 57.65%. The pruning ratios of Conv1, Conv3, Conv4 and Conv5 are 81.4%, 90.91%, 57.48%, and 57.53%, respectively. According to these pruning ratios, we prune each layer of the original model in a one-shot way. The pruned results are displayed in Table 5 and Table 6. Obviously, our pruning framework can achieve 76.42% and 23.08% reduction both on parameter and FLOPs, as well as maintain a strong representation ability. Especially, AFIE can achieve better results than the listed pruning results when the original model is only trained with one epoch, as well as achieve comparable results with other pruning methods when the model is fully trained. Therefore, our AFIE is an efficient criterion to help prune the under-trained AlexNet on MNIST.

Filters Pruning for VGG-16 on CIFAR-10 VGG-16 inherits the chain structure of AlexNet while extending the number of convolutional layers to 13. The original model achieves 93.35% top1-accuracy on CIFAR-10 with 150 training epochs. Similar to the previous experiment, we show the pruning ratio of each layer in Figure 2. As we can see, the overall pruning ratio λ^* is set as 65%, and Conv2

Epochs	Model	Methods	Par-O	FLOPs-O	Pru-R	Par-P	FLOPs-P	Top1-Acc
20	AlexNet	ThiNet (Luo, Wu, and Lin 2017)	3.52M	1.3×10^7	0.7	1.10M	1.0×10^7	0.9920
		DCP (Liu et al. 2022)				0.91M	1.0×10^7	0.9920
		SEP (Ding et al. 2021)				0.93M	1.0×10^7	0.9921
		l_1 (Li et al. 2017)				1.03M	1.0×10^7	0.9918
		Network Slimming (Liu et al. 2017)				1.16M	1.0×10^7	0.9921
		Taylor (Molchanov et al. 2017)				0.55M	1.0×10^7	0.9925
		AFIE				**0.83M**	$\mathbf{1.0\times10^7}$	**0.9920**
150	VGG-16	ThiNet (Luo, Wu, and Lin 2017)	33.65M	3.3×10^8	0.65	19.25M	1.6×10^8	0.9312
		DCP (Liu et al. 2022)				19.10M	1.3×10^8	0.9335
		SEP (Ding et al. 2021)				19.61M	1.0×10^8	0.9335
		l_1 (Li et al. 2017)				19.60M	1.6×10^8	0.9300
		Network Slimming (Liu et al. 2017)				19.39M	1.6×10^8	0.9335
		Taylor (Molchanov et al. 2017)				18.55M	1.0×10^8	0.9106
		AFIE				**19.15M**	$\mathbf{1.5\times10^8}$	**0.9335**
	ResNet-50	ThiNet (Luo, Wu, and Lin 2017)	25.56M	4.12×10^9	0.3	21.72M	3.56×10^9	0.758
		DCP (Liu et al. 2022)				21.91M	3.53×10^9	0.758
		SEP (Ding et al. 2021)				22.03M	3.52×10^9	0.758
		l_1 (Li et al. 2017)				22.23M	3.71×10^9	0.755
		Network Slimming (Liu et al. 2017)				21.95M	3.60×10^9	0.758
		Taylor (Molchanov et al. 2017)				20.85M	3.15×10^9	0.741
		AFIE				**21.88M**	$\mathbf{3.70\times10^9}$	**0.758**

Table 6: The comparison of different pruning methods for AlexNet, VGG-16, and ResNet-50 on MNIST, CIFAR-10, and ImageNet, respectively. The 20 epochs and 150 epochs mean AlexNet, VGG-16, and ResNet-50 are trained fully.

has the lowest pruning ratio with 13.84%. Conv1 is allotted with a 56.19% pruning ratio. Conv3, Conv4 and Conv5 are allocated with 27.68%, 23.66% and 47.37% pruning ratios, respectively. Conv6 and Conv7 are pruned with a 41.24% ratio. Finally, the remaining layers are assigned with the highest pruning ratio around 73.40%.

Then, one-shot pruning is conducted to prune the original model with the obtained pruning ratios both for the under-trained and fully-trained models. The results are displayed in Table 4, Table 5 and Table 6. Clearly, our AFIE pruning framework achieves impressive results with 43.09% and 54.55% reduction for parameters and FLOPs when the model is trained with 1 and 20 epochs respectively, and the accuracy also sacrifices tiny with under-trained status. Moreover, AFIE outperforms previous pruning methods with higher accuracy when the model is trained with 1 epoch, as well as achieves comparable compression results as other methods when the original model is fully-trained. This demonstrates that AFIE can prune VGG-16 at the early training stage, and then save much computational resources.

Filters Pruning for ResNet-50 on ImageNet Through the above experiments, we have shown the superiority of our AFIE pruning framework on AlexNet and VGG-16. Here, we further extend the work on ResNet-50 with more complicated structures. ResNet-50 is designed with 4 blocks and multiple skip connections. It has better representation ability on larger datasets like ImageNet. For this model, we only remove the filters of the second convolutional layers within each block with the size of 3×3 because they occupy the majority of FLOPs consumption. For simplicity, we rename the corresponding layers from Conv1 to Conv16. As shown in Table 4, the slimmed model can perform similar results as AlexNet and VGG-16, namely, the same reductions can be observed among under-trained and fully-trained models with similar accuracy.

When we set the overall pruning ratio λ^* to 30%, the pruning ratios of Conv1, Conv2, and Conv3 can be set to 17%, Conv4, Conv5, Conv6, and Conv7 can be set to 29%, Conv8, Conv9, Conv10, Conv11, Conv12, and Conv13 can be set to 51%, and Conv14, Conv15, and Conv16 can be specified as 90% according to Equation (10). As shown in Figure 2, we can see that the distribution of the pruning ratios within each block is flat, which means that the filters within the same block have similar importance. We perform one-shot pruning according to these ratios. The results are listed in Table 5 and Table 6. As shown, our method can outperform other methods when the model is trained with 1 epoch.

Conclusion

In this paper, we propose an entropy-based pruning framework AFIE to evaluate the importance of filters. It can yield stable evaluation for the filters no matter whether the original model is fully trained or poorly trained. We verify the pruning effectiveness of AFIE for CNN models of AlexNet, VGG-16, and ResNet-50, on datasets of MNIST, CIFAR-10, and ImageNet, respectively. The experimental results show that our method can effectively reduce the parameters and the FLOPs, as well as recover the prediction accuracy of the original model. A notable highlight is that our method can effectively perform pruning operations even when the original model is trained with only one epoch.

Acknowledgements

This work was supported in part by the National Natural Science Foundation of China under Grants 62133012, 61936006, 62073255, 62303366, and 62103314, in part by the Innovation Capability Support Program of Shaanxi under Grant 2021TD-05, and in part by the Key Research and Development Program of Shaanxi under Grant 2020ZDLGY04-07.

References

Alvarez, J. M.; and Salzmann, M. 2016. Learning the Number of Neurons in Deep Networks. In *Proceedings of Advances in Neural Information Processing Systems*, 2270–2278. Barcelona.

Chin, T.-W.; Zhang, C.; and Marculescu, D. 2018. Layer-Compensated Pruning for Resource-Constrained Convolutional Neural Networks. In *arXiv preprint arXiv:1810.00518*.

Ding, G.; Zhang, S.; Jia, Z.; Zhong, J.; and Han, J. 2021. Where to Prune: Using LSTM to Guide Data-Dependent Soft Pruning. *IEEE Transactions on Image Processing*, 30: 293–304.

Hassibi, B.; Stork, D.; wolff, G.; and Watanabe, T. 1993. Optimal Brain Surgeon. In *Proceedings of Advances in Neural Information Processing Systems*, 263–270. Denver, Colorado.

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2016. Deep Residual Learning for Image Recognition. In *Proceedings of 29th Conference on Computer Vision and Pattern Recognition*, 770–778. Las Vegas.

He, Y.; Kang, G.; Dong, X.; Fu, Y.; and Yang, Y. 2018. Soft Filter Pruning for Accelerating Deep Convolutional Neural networks. In *proceedings of International Joint Conference on Artificial Intelligence*, 2234–2240. Stockholm, Sweden.

He, Y.; Zhang, X.; and Sun, J. 2017. Channel Pruning for Accelerating very Deep Neural Networks. In *Proceedings of International Conference on Computer Vision*, 1398–1406. Venice, Italy.

Huang, Z.; Shao, W.; Wang, X.; Lin, L.; and Luo, P. 2021. Rethinking the Pruning Criteria for Convolutional Neural Network. In *Proceedings of Advances in Neural Information Processing Systems*. Online.

Huang, Z.; and Wang, N. 2018. Data-Driven Sparse Structure Selection for Deep Neural Networks. In *Proceedings of European Conference on Computer Vision.*, 317–334. Munich, Germany.

Jiang, C.; Li, G.; Qian, C.; and Tang, K. 2018. Efficient DNN Neuron Pruning by Minimizing Layer-wise Nonlinear Reconstruction Error. In *proceedings of International Joint Conference on Artificial Intelligence*, 2298–2304. Stockholm, Sweden.

Jonathan, F.; and Michael, C. 2019. The Lottery Ticket Hypothesis: Finding Sparse, Trainable Neural Networks. In *Proceedings of International Conference on Learning Representation*. New Orleans, Louisiana.

Krizhevsky, A.; Sutskever, I.; and Hinton, G. E. 2012. Imagenet Classification With Deep Convolutional Neural Networks. In *Proceedings of 26th Conference and Workshop on Neural Information Processing Systems*, 1106–1114. Lake Tahoe.

LeCun, Y.; Denker, J.; and Solla, S. 1990. Optimal Brain Damage. In *Proceedings of Advances in Neural Information Processing Systems*, 598–605. Denver, Colorado.

Li, H.; Kadav, A.; Durdanovic, I.; Samet, H.; and Graf, H. P. 2017. Pruning Filters for Efficient Convnets. In *Proceedings of International Conference on Learning Representation*. Toulon, France.

Liu, J.; Zhuang, B.; Zhuang, Z.; Guo, Y.; Huang, J.; Zhu, J.; and Tan, M. 2022. Discrimination-Aware Network Pruning for Deep Model Compression. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(8): 4035–4051.

Liu, Z.; Li, J.; Shen, Z.; Huang, G.; Yan, S.; and Zhang, C. 2017. Learning Efficient Convolutional Networks Through Network Slimming. In *Proceedings of International Conference on Computer Vision*, 2755–2763. Venice, Italy.

Liu, Z.; Mu, H.; Zhang, X.; Guo, Z.; Yang, X.; Cheng, K. T.; and Sun, J. 2019a. MetaPruning: Meta Learning For Automatic Neural Network Channel Pruning. In *Proceedings of International Conference on Computer Vision*, 3295–3304. Seoul, Korea.

Liu, Z.; Sun, M.; Zhou, T.; Huang, G.; and Darrell, T. 2019b. Rethinking the Value of Network Pruning. In *Proceedings of International Conference on Learning Representation*. New Orleans.

Luo, J.; Wu, J.; and Lin, W. 2017. ThiNet: A Filter Level Pruning Method for Deep Neural Network compression. In *Proceedings of International Conference on Computer Vision*, 5068–5076. Venice, Italy.

Mao, H.; Han, S.; Pool, J.; Li, W.; Liu, X.; Wang, Y.; and Dally, W. J. 2017. Exploring the Regularity of Sparse Structure in Convolutional Neural Networks. In *arXiv preprint arXiv:1705.08922*.

Molchanov, P.; Tyree, S.; Karras, T.; and Kautz, T. A. J. 2017. Pruning Convolutional Neural Networks for Resource Efficient Inference. In *Proceedings of International Conference on Learning Representation.*

Simonyan, K.; and Zissermann, A. 2015. Very Deep Convolutional Networks for Large-Scale Image Recognition. In *Proceedings of 3rd International Conference on Learning Representation*. San Diego.

Suau, X.; Zappella, L.; Palakkode, V.; and Apostoloff, N. 2018. Principal Filter Analysis for Guided Network Compression Convolutional Neural Networks. In *arXiv:1807.10585*.

Szegedy, C.; Liu, W.; Jia, Y.; Sermanet, P.; and Reed, S. 2015. Going Deeper with Convolutions. In *Proceedings of 28th Conference on Computer Vision and Pattern Recognition*, 770–778. Columbus, Ohio.

Ye, J.; Lu, X.; Lin, Z.; and Wang, J. Z. 2018. Rethinking The Smaller-Norm-Less-Informative Assumption in Channel Pruning of Convolutional Layers. In *Proceedings of International Conference on Learning Representation*. Vancouver, Canada.

Yu, R.; Li, A.; Chen, C.; Lai, J. H.; and Morariu, V. I. 2018. NISP: Pruning Networks Using Neuron Importance Score Propagation. In *Proceedings of Conference on Computer Vision and Pattern Recognition*, 9194–9203. Salt Lake City.

Zhang, X.; Zou, J.; He, K.; and Sun, J. 2016. Accelerating Very Deep Convolutional Networks for Classification and Detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 38(10): 1943–1955.

Zhou, H.; Alvarez, J.; and Porikli, F. 2016. Less Is More: Towards Compact CNNs. In *Proceedings of European Conference on Computer Vision*, 662–677. Amsterdan, Netherland.