Explainable Channel Pruning for Accelerating Deep Convolution Neural Networks via Class-wise Regularized Training*

Yuxin Zhang[1], Zhihua Wang[2], Xiaoyang Huang[1], Xiangjin Zhan[3], Yuqing Chang[4]

[1]Media Analytics and Computing Laboratory, School of Informatics, Xiamen University, China
[2]Medical Image Analysis Laboratory, School of Informatics, Xiamen University, China
[3]Big Data and Computational Intelligence Laboratory, School of Informatics, Xiamen University, China
[4]National Institute for Data Science in Health and Medicine, Xiamen University, China

Abstract

Channel Pruning has been long adopted for compressing CNNs, which significantly reduces the overall computation. Prior works implement channel pruning in an unexplainable manner, which tends to reduce the final classification errors while failing to consider the internal influence of each channel. In this paper, we conduct channel pruning in an explainable manner via class-wise regularized training, term as CRTPruner. Through deep visualization of feature maps activated by different channels, we observe that different channels have a varying contribution to different categories in image classification. Inspired by this, we choose to preserve channels contributing to most categories. Specifically, to model the contribution of each channel to differentiating categories, we develop a class-wise mask for each channel, implemented in a dynamic training manner w.r.t. the input image's category. On the basis of the learned class-wise mask, we perform a global voting mechanism to remove channels with less category discrimination. Lastly, a fine-tuning process is conducted to recover the performance of the pruned model. To our best knowledge, it is the first time that CNN interpretability theory is considered to guide channel pruning. Extensive experiments demonstrate the superiority of our CRTPruner over many state-of-the-arts. For instance, on CIFAR-10, it reduces 65.23% FLOPs with even 0.62% accuracy improvement for ResNet-110. On ILSVRC-2012, CRTPruner achieves a 45.6% FLOPs reduction with only a small loss of 0.83% in the top-1 accuracy for ResNet-50.

Introduction

Though convolutional neural networks (CNNs) have shown predominant performance in various computer vision tasks, such as image classification (Simonyan and Zisserman 2015; He et al. 2016), image super-resolution (Dong et al. 2015; Zhang et al. 2018), segmentation (He et al. 2017; Girshick 2015), and object detection (Girshick et al. 2014; Redmon et al. 2016), the vast demand on computation cost has prohibited them from being deployed on edge devices such as smartphones and embedded sensors. To address this, the researchers have developed several techniques for CNNs compression, such as network pruning (Han et al. 2015; He et al. 2020), parameter quantization (Hubara et al. 2016; Liu et al. 2020), tensor decomposition (Peng et al. 2018; Hayashi et al. 2019) and knowledge distillation (Romero et al. 2014; Hinton, Vinyals, and Dean 2015), *etc*. Among them, channel pruning has attracted ever increasing attention for its easy combination with general hardware and Basic Linear Algebra Subprograms (BLAS) libraries, which is thus the focus of this paper.

The core idea of channel pruning is to remove the entire channels in one filter so as to generate a sub-network of the original CNN with less computation cost. Existing studies could roughly be categorized into two categories. The first group abides a three-step pruning pipeline including pretraining a dense model, selection of "important" filters and fine-tuning the sub-net. Typically, most works of this category focus on the second step by either figuring out a filter importance estimation, such as ℓ_1-norm (Li et al. 2017), geometric information (He et al. 2019) and activation sparsity (Hu et al. 2016), or regarding channel pruning as an optimization problem (Lin et al. 2020; Guo, Ouyang, and Xu 2020). The second category implements channel pruning through retraining the network from scratch with additional sparsity constraints forced upon individual channels (Liu et al. 2017; Huang and Wang 2018; Luo and Wu 2020; Ding et al. 2020), after which, the pruned model can be available by removing zeroed channels or channels below a given threshold.

Though progress made in the past few years, existing methods build channel pruning on the basis of observing the CNN output, *i.e.*, the final classification performance, while leaving the internal influence of a CNN model hardly touched. For example, Li *et al*. (Li et al. 2017) removed filters with smaller ℓ_1-norms, which can indeed be viewed as to minimize the output difference between the original model and the pruned model. To take a more in-depth analysis, the massive non-linear operations inside CNNs make them hardly understandable. Thus, existing methods choose to regard the CNNs as a black box and observe the final output for network pruning. From this perspective, we term these methods "Black-Box pruning" in this paper.

Nevertheless, understanding the internal explanation of

*Project report for the 2020-2021 fall semester of deep learning course.

Figure 1: Visualization of various kinds of images (first row) along with the feature map of the 5-th and 144-th channel (second and third row, respectively) in the conv12 layer of VGG16-Net trained on ImageNet. (Best viewd with zooming in) All feature maps are with their origin activation value and size.

deep CNNs has attracted increasing attention (Wu et al. 2017; Yosinski et al. 2015; Zeiler and Fergus 2014; Zhang, Nian Wu, and Zhu 2018; Zhou et al. 2015), which also advances various vision tasks. For instance, Zeiler *et al.* (Zeiler and Fergus 2014) won the championship of the ILSVRC-2013 by adjusting architecture through visualization of internal feature maps. Inspired by this, we believe that exploring the internal logic in CNNs could be a promising prospect to guide channel pruning.

As exploited in (Yosinski et al. 2015), the feature maps of each channel have the locality that a particular area in one feature map is activated. Inspired by this, we visualize the feature maps generated by VGG16-Net (Simonyan and Zisserman 2015) trained on ImageNet to explore the local information in the internal layers of CNNs. As can be seen from Fig. 1, the 5-th channel at the 12-th convolution layer always generates feature maps that contain head information while the 144-th channel attempts to activate textual information. Even though there is no explicitly labeled head or text, this CNN model automatically learns to extract partial information to make better decisions, which exactly meets human intuition when classifying an image. That is, head information extracted by the 5-th channel helps the network to identify animals, and textual information extracted by the 144-th channel contributes to classify categories with texts such as digital watches. However, some local features may not be beneficial to identifying all categories. For example, the 144-th channel always chooses to deactivate most of the pixels when processing images with no textual semantics like dogs and pandas (see the third and fifth column in Fig. 1). Such local representation on the internal layers of a CNN shows that channels have a varying contribution to different categories in image classification, which motivates us to rethink the importance criterion of channel pruning.

Instead of simply considering the CNN output after removing a channel as prior arts do, we target at finding each channel's contribution to identifying different kinds of images. It is intuitive that if feature maps activated by one channel can benefit most categories' classification, this channel is essential and should be preserved; otherwise, it can be safely removed.

To this end, we assign each channel a class-wise mask, the length of which is basically the same as the category number in the training set. For each category of the input images, the corresponding mask is activated to multiply on the output feature map for model inference. By exerting a sparsity constraint that pulls the class-wise mask toward zero to counteract such gradients, these masks will maintain relatively large absolute values. Thus, after a few training epochs, each channel's importance score can be measured by the absolute sum of its class-wise mask, reflecting its overall contribution to identifying all categories. In this way, we can carry out the channel pruning in an explainable manner, for which we term our channel pruning as "CRTPruner". We further propose an iteratively global voting, which is performed using the above importance score to remove unimportant channels until the FLOPs of the pruned model meet pre-given computation budget. Lastly, a fine-tuning process is conducted to recover the performance of the pruned network.

Our contributions are summarized as follows:

- Based on an in-depth analysis of CNNs interpretation, we propose a novel explainable importance criteria for channel pruning that we should preserve channels beneficial to identifying most categories. To our best knowledge, it is the first time that CNN interpretability theory is considered to guide channel pruning.
- We carry out channel pruning in an explainable manner by jointly training a class-wise mask along with the original network to find each channel's contribution for classifying different categories, after which a global voting and a fine-tuning are conducted to obtain the final compact pruned model.
- Extensive experiments on CIFAR-10 and ILSVRC-2012 demonstrate the advantages of the proposed CRTPruner over several state-of-the-art advances in accelerating the CNNs.

Related Work

Channel Pruning. Channel pruning targets at snipping away entire channels in convolution kernel to obtain a pruned model, which not only saves computation cost, but is also compatible with off-the-shelf hardware. As discussed in Sec. , previous channel pruning works can be approximately divided into two groups. Starting from a pre-trained model, the first category designs various importance criteria to remove unimportant channels. For example, Li *et al*. (Li et al. 2017) chose to prune filters with smaller ℓ_1-norm. Molchanov *et al*. (Molchanov et al. 2016) proposed Taylor expansion to measure each channel's influence to the loss function as filter importance. In (Guo, Ouyang, and Xu 2020), Guo *et al*. considered both classification loss and feature importance as a pruning criterion to deal with the influence of next-layer feature map removal. The other group implements channel pruning in a training-adaptive manner by introducing extra sparsity regularization. For example, Huang *et al*. (Huang and Wang 2018) introduced

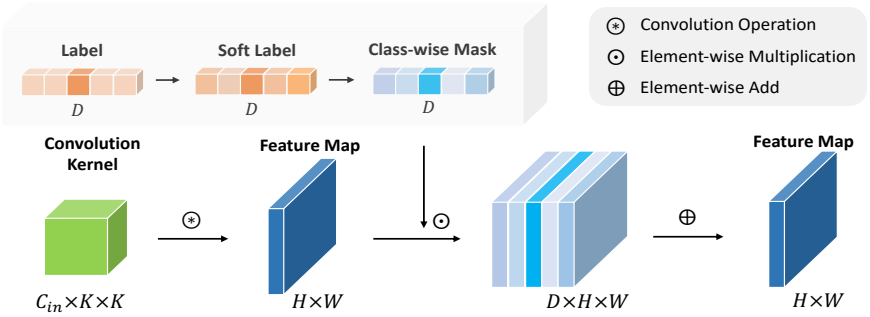

Figure 2: Framework of the proposed CRTPruner for class-wise mask training.(Best viewd with zooming in)

a scaling factor to scale the outputs of specific structures and added sparsity on these factors. They then trained the sparsity-regularized mask for network pruning through data-driven selection. Luo *et al*. (Luo and Wu 2020) employed an "autopruner" layer appended in the convolution layer to prune filters automatically. By regularizing auxiliary parameters instead of original weights values, Xiao *et al*. (Xiao, Wang, and Rajasekaran 2019) pruned the CNN model via a gradient-based updating rule. Both of these two groups conduct channel pruning with respect to the CNN output, failing to consider the internal mechanism of a CNN model. Though there are considerable improvements, interpretation for channel pruning remains an open problem.

Proposed Solution

Background

Considering an L-layer CNN model, its kernel weights can be represented as $\mathcal{W} = \{\mathcal{W}^1, \mathcal{W}^2, ..., \mathcal{W}^L\}$. The kernel in the l-th layer is denoted as $\mathcal{W}^l \in \mathbb{R}^{C^l_{out}\times C^l_{in}\times K^l\times K^l}$, where C^l_{out}, C^l_{in}, K^l denote the numbers of output channels and input channels, and the kernel size, respectively. Let $\mathcal{I}^l \in \mathbb{R}^{N\times C^l_{in}\times H^l\times W^l}$ be the input of the l-th layer where N is the batch size of input images, and H^l, W^l respectively stand for the height and width of the input. Given training image set X, associated with a set of class labels $Y^{N\times D}$ where D represents the total number of categories in the training set, we denote $X = \mathcal{I}^1$. For the i-th input image $X_{i,:,:,:}$, we treat its label $Y_{i,:}$ as a one-hot vector, that is $Y_{i,:} = j$ indicates that the j-th entry is set to one while the others are zero[1]. With a conventional CNN, the output of the l-th layer can be calculated as [2]

$$\mathcal{O}^l = \mathcal{I}^l \circledast \mathcal{W}^l, \tag{1}$$

where $\circledast$ denotes the convolutional operation. We have $\mathcal{O}^l = \mathcal{I}^{l+1} \in \mathbb{R}^{N\times C^{l+1}_{in}\times H^{l+1}\times W^{l+1}}$ and $C^l_{out} = C^{l+1}_{in}$. Then, its training loss can be expressed as

$$\min_{\mathcal{W}} \ \mathcal{L}(X, \mathcal{W}; Y). \tag{2}$$

[1]It also indicates that image $X_{i,:,:,:}$ belongs to the j-th class.

[2]The convolutional operation usually involves a bias term and is followed by a non-linear operation. For ease of representation, we omit them in this paper.

For channel pruning, a subgroup of output channels in $\mathcal{W}^l$ will be removed to obtain pruned kernel $\widehat{\mathcal{W}}^l \in \mathbb{R}^{\widehat{C}^l_{out}\times \widehat{C}^l_{in}\times K^l\times K^l}$ under the constraints of $\widehat{C}^l_{out} \leq C^l_{out}$ and $\widehat{C}^l_{in} \leq C^l_{in}$, thus it can reach a better trade-off between computation cost and accuracy performance. It is worth mentioning that the corresponding input channels of $\mathcal{W}^{l+1}$ are also removed. Accordingly, we can reformulate Eq. (1) and Eq. (2) in the pruned network as follows:

$$\widehat{\mathcal{O}}^l = \widehat{\mathcal{I}}^l \circledast \widehat{\mathcal{W}}^l, \tag{3}$$

$$\min_{\widehat{\mathcal{W}}} \ \mathcal{L}(X, \widehat{\mathcal{W}}; Y). \tag{4}$$

Class-wise Mask

The core of our CRTPruner is to assign per-layer kernel $\mathcal{W}^l$ a class-wise mask, which is formatted in the form of $\mathcal{M}^l \in \mathbb{R}^{D\times C_{out}}$. Specifically, the mask value $\mathcal{M}^l_{j,c}$ is built to measure the contributions of individual channels $\mathcal{W}^l_{c,:,:,:}$ to the network for recognizing the j-th category.

Then, for the i-th input image $X_{i,:,:,:}$ with label $Y_{i,:}$, the convolution using Eq. (1) in the forward propagation under our mask framework can be rewritten as

$$\mathcal{O}^l_{i,:,:,:} = \mathcal{I}^l_{i,:,:,:} \circledast (\mathcal{M}^l_{Y_{i,:}} * \mathcal{W}^l), \quad i = 1, 2, ..., N, \tag{5}$$

where $*$ denotes the channel-wise multiplication, that is, channel $\mathcal{W}^l_{c,:,:,:}$ is multiplied with the scalar mask $\mathcal{M}^l_{j,c}$. Subsequently, our mask-based training loss can be obtained as follows:

$$\min_{\mathcal{W},\mathcal{M}} \ \mathcal{L}(X, \mathcal{W}, \mathcal{M}; Y). \tag{6}$$

The rationale of our mask design lies in that, during back-propagation, the mask $\mathcal{M}^l_{j,c}$ will receive the gradient signals regarding the input images of the j-th category. On the premise of this principle, if channel $\mathcal{W}^l_{c,:,:,:}$ benefits the network to recognize input images from the j-th category, $\mathcal{M}^l_{j,c}$ will be positively activated, and deactivated, otherwise. Therefore, our class-wise mask design can well reflect the internal logic in CNNs, which seamlessly follows our motivation behind the channel pruning in our CRTPruner.

In comparison with typical CNNs where the label information is utilized in the loss layer, our class-wise mask-based convolutional operations are more label-guided since

it requires label information in every convolutional layer as shown in Eq. (5). This poses a critical challenge of the over-fitting problem since the label information is in the format of one-hot vector, meaning that we need to provide ground-truth labels for each convolution layer's forward propagation. Such data flow during training varies largely from the real testing part, thus may cause the over-fitting problem.

Inspired by the label-smoothing regularization (Szegedy et al. 2016), we propose to solve this problem by softening the one-hot vector, denoted as $\widehat{Y} \in \mathbb{R}^{N\times D}$, element of which is defined as

$$\widehat{Y}_{i,d} = \begin{cases} Y_{i,d} & \text{if } Y_{i,d} = 1, \\ \mathcal{N}(0,1) \cdot Y_{i,d} & \text{otherwise,} \end{cases} \tag{7}$$

where $\mathcal{N}(\cdot,\cdot)$ denotes the normal distribution.

Then, the convolution in Eq. (5) is reformulated as

$$\mathcal{O}^l_{i,:,:,:} = \mathcal{I}^l_{i,:,:,:} \circledast \Big(\big(\sum_{d=1}^{D} \widehat{Y}_{i,d} \cdot \mathcal{M}^l_{d,:} \big) * \mathcal{W}^l \Big), \quad i = 1, 2, ..., N. \tag{8}$$

As a result, channel pruning can be realized by removing those channels with poor masks, which indicate less contribution to object classification. To this end, it is natural that we can impose sparsity constraint on per-channel mask as

$$\min_{\mathcal{M}} \sum_{l=1}^{L} \sum_{c=1}^{C^l_{out}} \|\mathcal{M}^l_{:,c}\|_2. \tag{9}$$

The combination of our classification objective in Eq. (6) and sparsity constraint in Eq. (9) leads to the following final training loss:

$$\min_{\mathcal{W},\mathcal{M}} \mathcal{L}(X, \mathcal{W}, \mathcal{M}; Y) + \lambda \sum_{l=1}^{L} \sum_{c=1}^{C^l_{out}} \|\mathcal{M}^l_{:,c}\|_2. \tag{10}$$

Noticeably, the objective of Eq. (10) targets at locating channels that contribute more to recognizing the input images, which then makes up of the pruned kernel $\widehat{W}$ as described in Sec. , followed by a series of fine-tuning procedures using loss objective of Eq. (6). Therefore, only a few epochs are needed to train our class-wise mask so as to derive $\widehat{\mathcal{W}}$ in our empirical observation[3].

Global Voting for Cross-layer Pruning

Given a global pruning rate α, how to appropriately distribute it to each layer to preserve C^l_{out} channels would significantly affect the performance of the pruned model (Liu et al. 2019b). Prevalent methods resort to rule-of-thumb designs (Li et al. 2017; Lin et al. 2020) or complex structure search (Liu et al. 2019a; Li et al. 2020).

Fortunately, our CRTPruner can tacitly obtain a global important criterion for all channels in the network and conduct layer-wise pruning rate decision in an iteratively-voting manner. Detailedly, consider a trained class-wise mask $M^l_{:,c} \in \mathbb{R}^D$ of the c-th channel in the l-th layer, each item in this tensor represents this channel's ability for classifying one corresponding category of the dataset, thus we can measure this channel's contribution to overall classification performance by simply summing up these class-wise mask scores. We denote all scores of $\mathcal{M}^l$ as $\mathcal{S}^l \in \mathbb{R}^{C^l_{out}}$:

$$\mathcal{S}^l_c = \sum_{d=1}^{D} M^l_{d,c},\ c = 1\ ...\ C^l_{out}, \tag{11}$$

which then will serve as importance criterion for this channel.

Given a global pruning rate α, after obtaining all channels' importance scores $\mathcal{S}$ in the whole network, we iteratively remove the least-impact channels and calculate FLOPs pruning rate $\widehat{\alpha}$ of the current model until $\widehat{\alpha} \geq \alpha$. After voting, we integrate the left class-wise mask $\widehat{\mathcal{M}}$ into $\widehat{\mathcal{W}}$ to conduct fine-tuning for performance recovery. Particularly, as we soften the label obeying a standard normal distribution $\mathcal{N}(\mu = 0.5, \sigma = 1)$ during training except for the ground-truth related current input, the overall pruned $\widehat{\mathcal{M}}$ can be mixed into $\widehat{\mathcal{W}}$ by

$$\widehat{\mathcal{W}} = \widehat{\mathcal{W}} * \sum_{d=1}^{D} \mu\ \widehat{M_{d,:}}. \tag{12}$$

Lastly, we spend more epochs to fine-tune the pruned model for further performance recovery.

[3]We consider 10% of the total fine-tuning epochs for training the class-wise mask.

Table 1: Results for pruning ResNet-56 on CIFAR-10.

Model	Top-1 Acc.	Acc. ↓	FLOPs ↓
HRank (CVPR'20)	93.26% → 93.17%	0.09%	50.0%
SCP (ICML'20)	93.69% → 93.23%	0.46%	51.5%
SFP (IJCAI'18)	93.59% → 92.26%	1.33%	52.6%
LFPC (CVPR'20)	93.26% → 93.24%	0.02%	52.9%
DSA (ECCV'20)	93.12% → 92.91%	0.21%	53.2%
FPGM (CVPR'19)	93.59% → 92.93%	0.66%	53.6%
CRTPruner (Ours)	93.26% → **93.54%**	**-0.28%**	**55.6%**

Experiments

Implementation Details

We conduct extensive experiments on two representative datasets including CIFAR-10 (Krizhevsky, Hinton et al. 2009) and ILSVRC-2012 (Russakovsky et al. 2015) to demonstrate the efficacy of the proposed CRTPruner. We prune prevailing CNN models ResNet-56 (He et al. 2016) on CIFAR-10 and ResNet-50 (He et al. 2016) on ILSVRC-2012. We set the sparse parameter λ as 5×10^{-4} for all experiments. Then, We train our class-wise masks using the original full network with a learning rate of 0.1 for 30 epochs on CIFAR-10 and 9 epochs on ILSVRC-2012. After the global voting, the pruned model is then fine-tuned via the SGD optimizer. The momentum, batch size are set to 0.9, 256, respectively, in all experiments. On CIFAR-10, we iterate 300 epochs to fine-tune the pruned model with an initial learning rate of 0.1, that is divided by 10 at the 150-th and 225-th

Table 2: Results for pruning ResNet-50 on ILSVRC-2012.

Method	Top-1 Acc.	Top-1 Acc. ↓	Top-5 Acc.	Top-5 Acc. ↓	FLOPs	FLOPs↓
CP (ICCV'17)	76.15% → 72.30%	3.85%	92.96% → 90.80%	2.16%	2.73B	34.1%
SFP (IJCAI'18)	76.15% → 74.61%	1.54%	92.87% → 92.06%	0.81%	2.39B	41.8%
GAL (CVPR'19)	76.15% → 71.95%	4.20%	92.96% → 90.79%	2.17%	2.33B	43.7%
SSS-32 (ECCV'18)	76.12% → 71.82%	4.30%	92.86% → 90.79%	2.07%	2.33B	43.7%
HRank (CVPR'20)	76.15% → 75.01%	1.14%	92.96% → 92.33%	0.63%	2.30B	43.9%
CRTPruner (Ours)	76.15% → **75.32%**	**0.83%**	92.96% → **92.43%**	**0.53%**	**2.22B**	**45.6%**
FPGM (CVPR'19)	76.15% → 74.13%	2.02%	92.96% → 92.87%	0.09%	1.90B	53.5%
RRBP (CVPR'19)	76.15% → 73.00%	3.15%	92.96% → 91.00%	1.96%	1.86B	54.5%
ThiNet (ICCV '17)	72.88% → 71.01%	1.87%	91.06% → 90.02%	1.12%	1.71B	58.7%
LFPC (CVPR'19)	76.15% → 74.18%	1.97%	92.96% → 91.92%	1.04%	1.61B	60.8 %
HRank (CVPR'20)	76.15% → 71.98%	4.17%	92.96% → 91.01%	1.95%	1.55B	62.6%
CRTPruner (Ours)	76.15% → **74.21%**	**1.94%**	92.96% → **92.01%**	**0.95%**	**1.50B**	**63.5%**

epochs. On ILSVRC-2012, ResNet-50 is fine-tuned for 90 epochs with step scheduler learning rate, which begins at 0.1 and is divided by 10 every 30 epochs. All experiments are implemented with Pytorch (Paszke et al. 2019) and run on NVIDIA Tesla V100 GPUs.

Pruning ResNet-56 on CIFAR-10

We first demonstrate the superiority of CRTPruner on CIFAR-10 dataset. We evaluate the network pruning performances of various methods on ResNet (He et al. 2016),a predominant deep CNN with residual modules, as shown in Table 1. As can be observed, our CRTPruner increases the performance of original ResNet-56 by 0.28% and removes around 55.60% computation burden, while the other methods suffer the accuracy degradation more or less, even reducing less FLOPs.

Pruning ResNet-50 on ILSVRC-2012

We further demonstrate the efficacy of CRTPruner for pruning ResNet-50 (He et al. 2016) on the large-scale ILSVRC-2012. For fair comparison with many competitors , in Tab. 2, we list the performance of CRTPruner under similar pruning rates. Both the top-1 and top-5 accuracy along with FLOPs and FLOPs pruning rate of the pruned model are reported. As can be seen, in comparison with the SOTAs, CRTPruner shows the best performance under different pruning rates. Specifically, by setting the global pruning rate α to 0.45, CRTPruner reduces the FLOPs to around 2.22B while obtaining the top-1 accuracy of 75.32% and top-5 accuracy of 92.43%. In contrast, the recent SOTA, HRank (Lin et al. 2020), bears more computation of 2.30 FLOPs and poor top-1 accuracy of 75.01% and top-5 accuracy of 92.33%. Further, we increase pruning rate α to 0.63, where our CRTPruner shows the least accuracy drops of 2.02% in top-1 accuracy and 1.03% in top-5 accuracy. With less FLOPs reductions, LFPC (He et al. 2020) shows poor top-1 accuracy of 74.18% and top-5 accuracy of 91.92%. These results demonstrate the advantage of our CRTPruner for pruning the classic ResNet-50 on large-scale datasets.

Table 3: Top-1 accuracy comparison with/without class-wise mask for pruning ResNet-56 on CIFAR-10 under similar FLOPs pruning rate.

Setting	Top-1 Acc. ↓	FLOPs ↓
CRTPruner	**-0.28%**	**55.6%**
w/o Class-wise mask	1.43%	53.3%
w/o Soft mask	2.12%	54.7%

Ablation Study

Effect of Class-wise Mask. In this section, we prune ResNet-56 and test its performance on CIFAR-10 as an example to investigate the influences of individual components in the proposed class-wise mask. We first train each channel with a single mask for all categories of images, denoted as w/o Class-wise mask in Table 3. In addition, we conduct experiments without the smooth operation for mask activation, which is referred to as w/o Soft mask in Table 3. Our empirical observation shows that such an implementation leads to the over-fitting problem that the network will converge in one epoch. As a result, the trained mask cannot well contribute to discriminating different categories, which leads to an even worse top-1 accuracy drop than w/o Class-wise mask under a similar FLOPs pruning rate. Both experiments show remarkable efficacy of our class-wise mask.

Conclusion

Based on visualization and analysis of the deep feature in CNNs, we proposed a new perspective of channel pruning that one should preserve channels activating discriminative features for more categories in the dataset. We further carry out channel pruning in an explainable manner by devising a class-wise mask for each channel. During training, different sub-masks are activated for model inference, respecting to the current label of input images. A global voting and a fine-tuning are then performed to obtain the compressed model. Extensive experiments demonstrate the superiority of our new explainable perspective of network pruning. This work suggests new directions for more exploration of explainable neural network pruning.

References

Ding, X.; Hao, T.; Liu, J.; Han, J.; Guo, Y.; and Ding, G. 2020. Lossless CNN Channel Pruning via Gradient Resetting and Convolutional Re-parameterization. *arXiv preprint arXiv:2007.03260* .

Dong, C.; Loy, C. C.; He, K.; and Tang, X. 2015. Image super-resolution using deep convolutional networks. *TPAM)* 38(2): 295–307.

Girshick, R. 2015. Fast r-cnn. In *ICCV*, 1440–1448.

Girshick, R.; Donahue, J.; Darrell, T.; and Malik, J. 2014. Rich feature hierarchies for accurate object detection and semantic segmentation. In *CVPR*, 580–587.

Guo, J.; Ouyang, W.; and Xu, D. 2020. Channel pruning guided by classification loss and feature importance. In *AAAI*, 10885–10892.

Han, S.; Pool, J.; Tran, J.; and Dally, W. 2015. Learning both weights and connections for efficient neural network. In *NeurIPS*, 1135–1143.

Hayashi, K.; Yamaguchi, T.; Sugawara, Y.; and Maeda, S.-i. 2019. Exploring Unexplored Tensor Network Decompositions for Convolutional Neural Networks. In *NeurIPS*, 5552–5562.

He, K.; Gkioxari, G.; Dollar, P.; and Girshick, R. 2017. Mask R-CNN. In *CCV*.

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2016. Deep residual learning for image recognition. In *CVPR*, 770–778.

He, Y.; Ding, Y.; Liu, P.; Zhu, L.; Zhang, H.; and Yang, Y. 2020. Learning Filter Pruning Criteria for Deep Convolutional Neural Networks Acceleration. In *CVPR*, 2009–2018.

He, Y.; Liu, P.; Wang, Z.; Hu, Z.; and Yang, Y. 2019. Filter pruning via geometric median for deep convolutional neural networks acceleration. In *CVPR*, 4340–4349.

Hinton, G.; Vinyals, O.; and Dean, J. 2015. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531* .

Hu, H.; Peng, R.; Tai, Y.-W.; and Tang, C.-K. 2016. Network trimming: A data-driven neuron pruning approach towards efficient deep architectures. *arXiv preprint arXiv:1607.03250* .

Huang, Z.; and Wang, N. 2018. Data-driven sparse structure selection for deep neural networks. In *ECCV*, 304–320.

Hubara, I.; Courbariaux, M.; Soudry, D.; El-Yaniv, R.; and Bengio, Y. 2016. Binarized neural networks. In *NeurIPS*, 4107–4115.

Krizhevsky, A.; Hinton, G.; et al. 2009. Learning multiple layers of features from tiny images .

Li, B.; Wu, B.; Su, J.; Wang, G.; and Lin, L. 2020. EagleEye: Fast Sub-net Evaluation for Efficient Neural Network Pruning. *arXiv preprint arXiv:2007.02491* .

Li, H.; Kadav, A.; Durdanovic, I.; Samet, H.; and Graf, H. P. 2017. Pruning filters for efficient convnets. In *ICLR*.

Lin, M.; Ji, R.; Wang, Y.; Zhang, Y.; Zhang, B.; Tian, Y.; and Shao, L. 2020. HRank: Filter Pruning using High-Rank Feature Map. In *CVPR*, 1529–1538.

Liu, Z.; Li, J.; Shen, Z.; Huang, G.; Yan, S.; and Zhang, C. 2017. Learning efficient convolutional networks through network slimming. In *ICCV*, 2736–2744.

Liu, Z.; Luo, W.; Wu, B.; Yang, X.; Liu, W.; and Cheng, K.-T. 2020. Bi-real net: Binarizing deep network towards real-network performance. *IJCV* 128(1): 202–219.

Liu, Z.; Mu, H.; Zhang, X.; Guo, Z.; Yang, X.; Cheng, T. K.-T.; and Sun, J. 2019a. MetaPruning: Meta Learning for Automatic Neural Network Channel Pruning. In *ICCV*, 3296–3305.

Liu, Z.; Sun, M.; Zhou, T.; Huang, G.; and Darrell, T. 2019b. Rethinking the value of network pruning. In *ICLR*.

Luo, J.-H.; and Wu, J. 2020. Autopruner: An end-to-end trainable filter pruning method for efficient deep model inference. *PR* 107461.

Molchanov, P.; Tyree, S.; Karras, T.; Aila, T.; and Kautz, J. 2016. Pruning convolutional neural networks for resource efficient inference. *arXiv preprint arXiv:1611.06440* .

Paszke, A.; Gross, S.; Massa, F.; Lerer, A.; Bradbury, J.; Chanan, G.; Killeen, T.; Lin, Z.; Gimelshein, N.; Antiga, L.; et al. 2019. Pytorch: An imperative style, high-performance deep learning library. In *NeurIPS*, 8026–8037.

Peng, B.; Tan, W.; Li, Z.; Zhang, S.; Xie, D.; and Pu, S. 2018. Extreme network compression via filter group approximation. In *ECCV*, 300–316.

Redmon, J.; Divvala, S.; Girshick, R.; and Farhadi, A. 2016. You only look once: Unified, real-time object detection. In *CVPR*, 779–788.

Romero, A.; Ballas, N.; Kahou, S. E.; Chassang, A.; Gatta, C.; and Bengio, Y. 2014. Fitnets: Hints for thin deep nets. *arXiv preprint arXiv:1412.6550* .

Russakovsky, O.; Deng, J.; Su, H.; Krause, J.; Satheesh, S.; Ma, S.; Huang, Z.; Karpathy, A.; Khosla, A.; Bernstein, M.; et al. 2015. Imagenet large scale visual recognition challenge. *IJCV* 115(3): 211–252.

Simonyan, K.; and Zisserman, A. 2015. Very Deep Convolutional Networks for Large-Scale Image Recognition. In *ICLR*.

Szegedy, C.; Vanhoucke, V.; Ioffe, S.; Shlens, J.; and Wojna, Z. 2016. Rethinking the inception architecture for computer vision. In *CVPR*, 2818–2826.

Wu, T.; Li, X.; Song, X.; Sun, W.; Dong, L.; and Li, B. 2017. Interpretable r-cnn. *arXiv preprint arXiv:1711.05226* 2.

Xiao, X.; Wang, Z.; and Rajasekaran, S. 2019. Autoprune: Automatic network pruning by regularizing auxiliary parameters. In *NeurIPS*, 13681–13691.

Yosinski, J.; Clune, J.; Nguyen, A.; Fuchs, T.; and Lipson, H. 2015. Understanding neural networks through deep visualization. In *ICML*.

Zeiler, M. D.; and Fergus, R. 2014. Visualizing and understanding convolutional networks. In *ECCV*, 818–833. Springer.

Zhang, Q.; Nian Wu, Y.; and Zhu, S.-C. 2018. Interpretable convolutional neural networks. In *CVPR*, 8827–8836.

Zhang, Y.; Li, K.; Li, K.; Wang, L.; Zhong, B.; and Fu, Y. 2018. Image super-resolution using very deep residual channel attention networks. In *ECCV*, 286–301.

Zhou, B.; Khosla, A.; Lapedriza, À.; Oliva, A.; and Torralba, A. 2015. Object Detectors Emerge in Deep Scene CNNs. In *ICLR*.