ACP: ADAPTIVE CHANNEL PRUNING FOR EFFICIENT NEURAL NETWORKS

Yuan Zhang[1,2], *Yuan Yuan*[2], *Qi Wang*[1,2*]

[1]Unmanned System Research Institute,
[2]School of Artificial Intelligence, Optics and Electronics (iOPEN),
Northwestern Polytechnical University, Xi'an 710072, Shaanxi, P.R. China.

ABSTRACT

In recent years, deep convolutional neural networks have achieved amazing results on multiple tasks. However, these complex network models often require significant computation resources and energy costs, so that they are difficult to deploy to power-constrained devices, such as IoT systems, mobile phones, embedded devices, etc. Aforementioned challenges can be overcome through model compression like network pruning. In this paper, we propose an adaptive channel pruning module (ACPM) to automatically adjust the pruning rate with respect to each channel, which is more efficient to prune redundant channel parameters, as well as more robust to datasets and backbones. With one-shot pruning strategy design, the model compression time can be saved significantly. Extensive experiments demonstrate that ACPM makes tremendous improvement on both pruning rate and accuracy, and also achieves the state-of-the-art results on a series of different networks and benchmarks.

Index Terms— Efficient deep learning, model compression, channel pruning, neural network acceleration

1. INTRODUCTION

Recently, convolutional neural networks (CNNs) have made a huge impact on vision tasks, which become a dominant approach in deep learning. Typical applications of CNNs include object detection [1, 2, 3], image classification [4, 5, 6] and segmentation [7, 8]. However, their high demands for memory usage and computing resources prohibit the vast majority of state-of-the-art CNNs from being deployed on the hardware devices, which are computing power and storage constrained, such as smartphones, embedded devices, etc. For example, ResNet-152 [4], which has 152 layers and more than 60 million parameters, requires 12 Giga float-point-operations (FLOPs) during a single inference. Even the smallest ResNet, ResNet-18 also requires 2 GFLOPs, which is unlikely to be affordable on resource constrained platforms.

*Qi Wang is the corresponding author. This work was supported by the National Natural Science Foundation of China under Grant U21B2041, U1864204, 61632018, and 61825603.

Fig. 1. The procedure of our one-shot channel pruning method, which is more efficient to compress the CNNs and finally obtains a lightweight network.

To solve the aforementioned problems, efficient model compression and acceleration techniques are designed to reduce the redundancy parameters of deep CNNs, which make a better trade-off between model efficiency and prediction accuracy. Popular model compression techniques mainly contain three categories, namely, knowledge distillation [11, 12], network pruning [13, 14] and parameter quantization [9, 10].

Among them, network pruning is a method that can systematically remove redundant parameters in the original large neural network, showing great potential in various practical applications. Pruning has been used since the late 1990s [13, 14], however, due to the rise of convolutional neural networks, explosive research interests have recently ushered in. Typical works either make the weights of filters sparse in CNNs (weight pruning) [15, 16], or remove entire redundant channels in each layer and the corresponding parameters (channel pruning) [17, 18, 19, 20, 21]. Weight pruning is a method that stores a good deal of indexes to achieve high-efficiency acceleration, which is unstructured and therefore requires special computing platform support. In contrast, channel pruning methods are hardware-friendly since the entire channels are removed, producing a non-sparse compressed model. In this paper, we are mainly concerned with channel pruning to achieve efficient network compression.

Recently, [17] proposes a L1-norm based method, which assumes the channels with small L1-norm can be pruned safely due to there less information content, and then regain accuracy by fine-tuning the network. In article [20], the importance of channels is evaluated by the first-order gradient, and the least important ones will be removed. LASSO-based channel selection strategy is proposed in [18], and through feature map reconstruction error minimization, the redundant channels will be pruned. In [21], L1 regularization is imposed on the BN scaling factors and the redundant channels will be

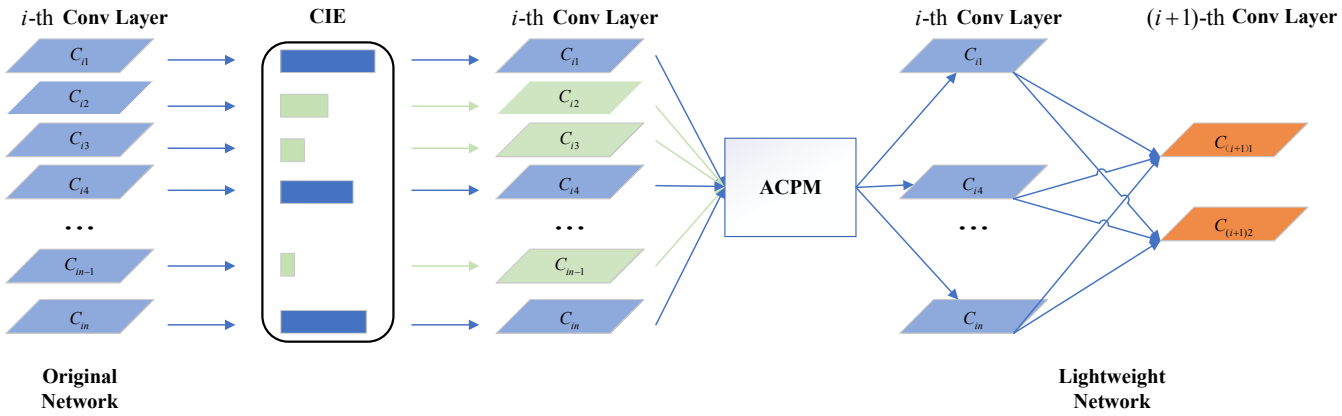

Fig. 2. The overall pipeline of the proposed ACP. First, the Channel Importance Evaluation (CIE) is used to evaluate the channels' importance in the i-th layer of original network. After that, the scaling factor values of each channel are obtained. Then, using the adaptive channel pruning module (ACPM), the corresponding pruning rate of each layer is adaptively learned, according to its importance and the global pruning rate constraint. Finally, the redundant channel can be pruned safely with little accuracy loss and obtain lightweight network.

pruned iteratively by the two-step method, which called Network Slimming. In article [19], HRank explores specific filter's feature map rank, therefore, those with low-rank feature maps that contain less information should be pruned. However, there are some drawbacks to these methods. At first, the prune-retrain cycle is adopted in [19], which is time-consuming because the redundancy of channels needs to be re-evaluated repeatedly and retrain the pruned network at each time. Secondly, it is very expensive to train CNNs from scratch [21] when sparsity regularization is induced. Thirdly, these methods [17, 20, 18] can not adaptively determine the channel pruning rate of each layer and automatically adjust pruning strategy with regard to each channel, thus leading to sub-optimal results.

In this paper, we propose an adaptive channel pruning method called ACP for network pruning. Our method meets the needs of the aforementioned problems through adapting the channel pruning rate of each layer and the one-shot pruning design. Specifically, the pruning rate adaptiveness is realized by using the pruning accuracy drop estimation step through each channel as well as the global pruning rate designed according to computation cost constrains. Moreover, the proposed ACP improves channel pruning results, with no more performance sacrifices, even achieves higher accuracy than the original unpruned network in many cases. Our main contributions are three-folds:

1. A novel method called ACP, which can determine the channel pruning rate of each layer adaptively, is proposed.

2. Our method is one-shot that doesn't need to adopt prune-retrain cycle, which is more time-saving and efficient.

3. Extensive experiments on mainstream dataset CIFAR-10 and more challenging dataset CIFAR-100 demonstrate that our approach exceeds state-of-the-art methods.

2. PROPOSED METHODOLOGY

2.1. Preliminaries

For a given convolutional neural network model M with trainable parameters $W = \{W_1, W_2, ..., W_L\}$ and L layers, it is assumed that its size is S, and the number of FLOPs is F. Given a dataset $D = \{(x_1, y_1), ..., (x_N, y_N)\}$ contains N input-output pairs, the accuracy on D is defined as acc.

After pruning, the corresponding parameters are M', S', F', and acc' respectively. The purpose of pruning is to remove redundant parameters while minimizing the loss of accuracy. The model after pruning needs to reduce $ds\%$ of original network size or $df\%$ FLOPs. Therefore, the optimization goal of the pruning task can be defined as follows:

$$f(M', D, M) = \min_{M'}(acc(M, D) - acc'(M', D)),$$
$$s.t. \begin{cases} \frac{S(M)-S'(M')}{S(M)} \geq ds\% \\ \frac{F(M)-F'(M')}{F(M)} \geq df\%. \end{cases} \quad (1)$$

2.2. Channel Importance Evaluation (CIE)

Batch Normalization (BN) is first proposed in [22], which has been adopted by a mass of modern CNNs as an essential component. With BN, previous layer's output can be normalized to adjust the range, thus achieving fast and stable convergence for networks. Let X and Y be the input and output of a BN layer, the definition of BN is as follows:

$$Y = \gamma \frac{X - \mu_\kappa}{\sqrt{\sigma_\kappa^2 + \varepsilon}} + \beta, \quad (2)$$

where κ denotes the mini-batch in current iteration, σ_κ is the standard deviation and μ_κ is the mean of the current mini-

batch input κ, respectively. γ and β are the transformation parameters which can be trained, representing scale and shift operations. In this way, activations will be linearly transformed back to any scale.

The weight of BN with traditional training method is often not too close to 0, or is relatively large. In this case, directly pruning the channel with a relatively small weight will have a greater impact on the model efficiency.

Therefore, inspired by [21], we introduce sparse training. By adding the L1 regularization to the γ parameters in BN layers, which regards as the scaling factors, and force the BN parameters to move closer to 0 during the training process. After training is completed, the part of the BN parameter closest to 0 corresponds to the pruning channel, i.e., cutting off the least important channel. In this way, we can simply evaluate channel importance of each layer. The overall loss function is defined as follows:

$$Loss = \sum_{(x,y)} l(y', y) + \lambda \|\gamma\|_1, \tag{3}$$

$$y' = f_D(x, W), \tag{4}$$

where y' denotes the predicting result on dataset D, with input x and trainable parameters W. (x, y) is training input and label pair, λ balances the training and sparsity. Note that the first sum item is a common classification loss.

Network Slimming [21] chooses a percentile threshold manually and channels with γ smaller than the set threshold will be pruned, which may prune some important channels and cause a significant accuracy decline. Different with [21], our Adaptive Channel Pruning Module can adaptively make a decision of the channel pruning rate of each layer, to some extend, avoiding over-pruning some important layers.

2.3. Adaptive Channel Pruning Module (ACPM)

For a given network with L layers and L_i has n_i channels denoted as $L_i = \{C_1, C_2, ..., C_{n_i}\}$, we assume the pruning rate of L_i is denoted as p_i, and the global pruning rate is p_g. The constraint relationship between the two is simply defined as follows:

$$\sum_{i=1}^{L} S_i p_i \geq S p_g, \tag{5}$$

$$\sum_{i}^{L} p_i = p_g, \tag{6}$$

where S_i denotes the parameter quantity in layer i, and the total number of parameters in M is S, as defined in Equation (1).

Inspired by [23], we assume the error tolerance parameter as η while pruning, and the redundant channels will be pruned by ACPM only when the accuracy drop is within η. After CIE, the channel importance parameter γ_i of L_i can be obtained, then they are sorted in ascending order, and the channels with small γ can be regard as less important channels. Further, ACPM prunes the channels within η iteratively, and computing the accuracy loss in each iteration. In this way, the channel pruning rate of each layer can gradually grow larger by adding αp_i, where α is a scaling factor, which is small. In the end, the optimal channel pruning rate can be obtained with the least accuracy drop or even higher than the original network. The detailed algorithm description is shown in Algorithm 1.

Algorithm 1 Adaptive Channel Pruning Module (ACPM)

Input: CIE of each layer: γ_i, Global pruning rate: p_g, Scaling factor: α

Parameters: Error tolerance: η, Number of parameters: S_i, Total number of parameters: S, Accuracy of current model: acc

Output: The pruning rate of L_i: p_i

```
Sorting γ_i in ascending order
Initialize η, p_i
while (acc(M) − acc(M')) ≤ η and sum(S_i p_i) ≥ S p_g do
    for i = 1 → L do
        γ_i = γ_i − p_i γ_i
        if (acc(M) − acc(M')) > η then
            break
        else
            p_i = p_i + α p_i
        end if
        return p_i
    end for
end while
```

3. EXPERIMENTS AND RESULTS

In this Section, we conduct dozens of experiments on two mainstream classification datasets: CIFAR-10 and CIFAR-100 [24]. We compare the proposed ACP with other SOTA channel pruning methods to show the effectiveness and efficiency of our method.

3.1. Implementation Details

VGG-16 [5], ResNet-56 [4] and ResNet-110 are pruned on CIFAR datasets. The CIFAR-10 dataset consists of 50,000 images for training and 10,000 images for testing, respectively. All images are with resolution 32x32 in 10 classes. While CIFAR-100 dataset has 100 classes, each class contains 600 images with 500 training images and 100 test images. Following the settings in the article [21], the baseline model is trained for 160 epochs. The batch size and initial learning rate are set to 64 and 0.1, respectively. The learning rate will be divided by 10 at 50% and 75% of total training epochs. Using the SGD optimizer with momentum set to 0.9

Table 1. Pruning results on CIFAR-10 dataset. Accu. means the test accuracy on dataset and PR denotes the pruning rate. Note that our method is denoted as ACP-p_g, where p_g is the global pruning rate.

Model	Accu.(%)	Parameters(PR)	FLOPs(PR)
VGG-16	93.73	14.72M(0%)	314.16M(0%)
L1[17]	93.40	5.40M(64.00%)	206.00M(34.30%)
VCNNP[25]	93.18	3.92M(73.34%)	190.00M(39.10%)
HRank[19]	93.43	2.51M(82.95%)	145.61M(53.65%)
ACP-0.68	**93.78**	**1.92M(86.96%)**	**74.39M(76.32%)**
HRank[19]	91.23	1.78M(87.91%)	73.70M(76.54%)
GT[26]	93.27	0.65M(95.58%)	70.20M(77.65%)
ResNet-56	93.92	0.85M(0%)	126.58M(0%)
L1[17]	93.06	0.73M(14.10%)	90.90M(27.60%)
HRank[19]	93.52	0.71M(16.47%)	88.72M(29.90%)
GT[26]	**94.58**	**0.56M(34.12%)**	**79.75M(37.00%)**
ACP-0.4	94.06	0.52M(38.82%)	73.12M(42.23%)
ACP-0.5	93.88	0.49M(42.35%)	67.63M(46.57%)
ACP-0.6	92.56	0.43M(49.41%)	58.66M(53.65%)
ResNet-110	94.47	1.73M(0%)	255.01M(0%)
L1[17]	93.30	1.16M(32.60%)	155.00M(38.70%)
HRank[19]	94.23	1.04M(39.40%)	148.70M(41.20%)
ACP-0.4	95.03	0.98M(43.35%)	138.41M(45.72%)
ACP-0.5	94.68	0.91M(47.40%)	126.96M(50.21%)
ACP-0.6	**94.45**	**0.81M(53.18%)**	**110.39M(56.71%)**
GT[26]	94.39	0.79M(54.34%)	128.78M(49.50%)

and weight decay set to 10^{-4}. For VGG-16 sparse training, λ is set to 10^{-4} and for ResNet is 10^{-5}. During the pruning stage, p_i is set to 0.01 initially and will change adaptively. While the scaling factor α and error tolerance η are set to 0.1 and 0.005 respectively. The optimization settings in the fine-tuning are the same as in training.

3.2. Results on CIFAR-10

The channel pruning results of VGG-16, ResNet-56 and ResNet-110 on CIFAR-10 dataset are illustrated in Table 1. Compared with the SOTA pruning methods: L1 [17], VCNNP [25], HRank [19] and GT [26], the proposed ACP exhibits excellent performance. For example, when pruned 86.96% parameters and 76.32% FLOPs on VGG-16 with a global pruning rate 0.68, the accuracy is even higher than the original network (93.78% vs. 93.73%), which is significantly better than the previous methods such as L1, VCNNP, HRank and GT. The reason may be that redundant parameters are pruned appropriately, the model's generalization ability is improved and thus alleviate overfitting. This phenomenon has also occurred on pruned ResNet-56 and ResNet-110.

For ResNet-110, when pruned 53.18% parameters and 56.71% FLOPs, the test accuracy decreases by only 0.02%, which further shows that our proposed method is quite competitive.

Table 2. Pruning results on CIFAR-100 dataset. Mainly compared with GT and original network.

Model	Accu.(%)	Parameters(PR)	FLOPs(PR)
VGG-16	71.13	14.72M(0%)	314.16M(0%)
ACP-0.38	**72.87**	**5.27M(64.20%)**	**152.77M(51.37%)**
GT[26]	70.91	1.99M(86.48%)	99.97M(68.18%)
ResNet-56	74.06	0.85M(0%)	126.58M(0%)
ACP-0.4	74.52	0.53M(37.65%)	75.78M(40.13%)
ACP-0.5	**74.36**	**0.50M(41.18%)**	**69.78M(44.87%)**
ACP-0.6	73.31	0.45M(47.06%)	61.23M(51.62%)
ResNet-110	75.79	1.73M(0%)	255.01M(0%)
ACP-0.4	76.65	1.01M(41.62%)	144.97M(43.15%)
ACP-0.5	**76.52**	**0.95M(45.09%)**	**131.73M(48.34%)**
ACP-0.6	74.39	0.86M(50.29%)	116.61M(54.27%)

3.3. Results on CIFAR-100

The channel pruning results of VGG-16, ResNet-56 and ResNet-110 on CIFAR-100 dataset are illustrated in Table 2. Since there are few pruning methods report results on CIFAR-100, we mainly compare our pruning results with GT and the original model. For VGG-16, when pruned 64.20% parameters and 51.37% FLOPs, the test accuracy outperforms GT (72.87 vs. 70.91%), and even 1.74% higher than the original network. For ResNet-56 pruned 41.18% parameters, 69.78% FLOPs and for ResNet-110 pruned 45.09% parameters, 48.34% FLOPs, the similar phenomenon happen again.

From the pruning results, we further observe that the parameters and FLOPs saving on ResNets are less significant than VGGNet, the PR is slightly lower than the global pruning rate. We guess the reason is ResNets have fewer redundant parameters than VGGNet, due to the effect of the "Bottleneck" structure.

4. CONCLUSION

In this paper, we propose an adaptive channel pruning method called ACP for network pruning. Our method adaptively determines the channel pruning rate of each layer, which enables optimal pruning rate and reach higher compression performance. With one-shot pruning strategy design, the model compression time can be saved significantly. Extensive experiments on two mainstream datasets CIFAR-10 and CIFAR-100 show that our method exceeds state-of-the-art methods. In the future work, we will try to apply ACP to lightweight networks like MobileNet and EfficientNet $et\ al.$, which is currently a challenge in the network pruning field.

5. REFERENCES

[1] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," *NeurIPS*, vol. 28, pp. 91–99, 2015.

[2] Yuan Yuan, Zhitong Xiong, and Qi Wang, "Vssa-net: vertical spatial sequence attention network for traffic sign detection," *T-IP*, vol. 28, no. 7, pp. 3423–3434, 2019.

[3] Qi Wang, Tao Han, Zequn Qin, Junyu Gao, and Xuelong Li, "Multitask attention network for lane detection and fitting," *T-NNLS*, 2020.

[4] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun, "Deep residual learning for image recognition," in *CVPR*, 2016, pp. 770–778.

[5] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *ICLR*, 2015.

[6] Qi Wang, Wei Huang, Zhitong Xiong, and Xuelong Li, "Looking closer at the scene: Multiscale representation learning for remote sensing image scene classification," *T-NNLS*, pp. 1–15, 2020.

[7] Jonathan Long, Evan Shelhamer, and Trevor Darrell, "Fully convolutional networks for semantic segmentation," in *CVPR*, 2015, pp. 3431–3440.

[8] Yuan Yuan, Jie Fang, Xiaoqiang Lu, and Yachuang Feng, "Spatial structure preserving feature pyramid network for semantic image segmentation," *TOMM*, vol. 15, no. 3, pp. 1–19, 2019.

[9] Qi Wang, Nianhui Guo, Zhitong Xiong, Zeping Yin, and Xuelong Li, "Gradient matters: Designing binarized neural networks via enhanced information-flow," *T-PAMI*, 2021.

[10] Zhenhua Liu, Xinfeng Zhang, Shanshe Wang, Siwei Ma, and Wen Gao, "Evolutionary quantization of neural networks with mixed-precision," in *ICASSP*, 2021, pp. 2785–2789.

[11] Dong Liang, Yun Du, Han Sun, Liyan Zhang, Ningzhong Liu, and Mingqiang Wei, "Nlkd: Using coarse annotations for semantic segmentation based on knowledge distillation," in *ICASSP*, 2021, pp. 2335–2339.

[12] Mingi Ji, Byeongho Heo, and Sungrae Park, "Show, attend and distill: Knowledge distillation via attention-based feature matching," in *AAAI*, 2021, vol. 35, pp. 7945–7952.

[13] Yann LeCun, John S Denker, and Sara A Solla, "Optimal brain damage," in *NeurIPS*, 1990, pp. 598–605.

[14] Babak Hassibi and David G Stork, "Second order derivatives for network pruning: Optimal brain surgeon," *NeurIPS*, pp. 164–171, 1992.

[15] Song Han, Jeff Pool, John Tran, and William J. Dally, "Learning both weights and connections for efficient neural network," in *NeurIPS*, 2015, pp. 1135–1143.

[16] Miguel A Carreira-Perpinán and Yerlan Idelbayev, ""learning-compression" algorithms for neural net pruning," in *CVPR*, 2018, pp. 8532–8541.

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

[18] Yihui He, Xiangyu Zhang, and Jian Sun, "Channel pruning for accelerating very deep neural networks," in *ICCV*, 2017, pp. 1389–1397.

[19] Mingbao Lin, Rongrong Ji, Yan Wang, Yichen Zhang, Baochang Zhang, Yonghong Tian, and Ling Shao, "Hrank: Filter pruning using high-rank feature map," in *CVPR*, 2020, pp. 1529–1538.

[20] Pavlo Molchanov, Stephen Tyree, Tero Karras, Timo Aila, and Jan Kautz, "Pruning convolutional neural networks for resource efficient inference," *arXiv preprint arXiv:1611.06440*, 2016.

[21] Zhuang Liu, Jianguo Li, Zhiqiang Shen, Gao Huang, Shoumeng Yan, and Changshui Zhang, "Learning efficient convolutional networks through network slimming," in *ICCV*, 2017, pp. 2736–2744.

[22] Sergey Ioffe and Christian Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *ICML*, 2015, pp. 448–456.

[23] Pravendra Singh, Vinay Kumar Verma, Piyush Rai, and Vinay P. Namboodiri, "Play and prune: Adaptive filter pruning for deep model compression," in *IJCAI*, 2019, pp. 3460–3466.

[24] Alex Krizhevsky and Geoffrey Hinton, "Learning multiple layers of features from tiny images," 2009.

[25] Chenglong Zhao, Bingbing Ni, Jian Zhang, Qiwei Zhao, Wenjun Zhang, and Qi Tian, "Variational convolutional neural network pruning," in *CVPR*, 2019, pp. 2780–2789.

[26] Fang Yu, Chuanqi Han, Pengcheng Wang, Xi Huang, and Li Cui, "Gate trimming: One-shot channel pruning for efficient convolutional neural networks," in *ICASSP*, 2021, pp. 1365–1369.