On the Channel Pruning Using Graph Convolution Network for Convolutional Neural Network Acceleration

Di Jiang, Yuan Cao and **Qiang Yang**
College of Electrical Engineering, Zhejiang University, 310027, Hangzhou China
qyang@zju.edu.com

Abstract

Network pruning is considered efficient for sparsification and acceleration of Convolutional Neural Network (CNN) based models that can be adopted in resource-constrained environments. Inspired by two popular pruning criteria, i.e. magnitude and similarity, this paper proposes a novel structural pruning method based on Graph Convolution Network (GCN) to further promote compression performance. The channel features are firstly extracted by Global Average Pooling (GAP) from a batch of samples, and a graph model for each layer is generated based on the similarity of features. A set of agents for individual CNN layers are implemented by GCN and utilized to synthesize comprehensive channel information and determine the pruning scheme for the overall CNN model. The training process of each agent is carried out using Reinforcement Learning (RL) to ensure their convergence and adaptability to various network architectures. The proposed solution is assessed based on a range of image classification datasets i.e., CIFAR and Tiny-ImageNet. The numerical results indicate that the proposed pruning method outperforms the pure magnitude-based or similarity-based pruning solutions and other SOTA methods (e.g., HRank and SCP). For example, the proposed method can prune VGG16 by removing 93% of the model parameters without any accuracy reduction in the CIFAR10 dataset.

1 Introduction

In recent years, the Convolutional Neural Networks (CNNs) have been widely adopted in many application domains, e.g., computer vision. However, the requirements of substantial storage space and computing power make such models difficult to be deployed in computing and storage constrained platforms. Model compression is one of the feasible solutions to address this challenge through designing compact and efficient models whilst maintaining the accuracy as much as possible. Pruning based on a specific criterion is an effective way to compress the model, and the most commonly used criteria are magnitude and similarity. The magnitude-based methods prefer to prune the components with smaller magnitude in the network. For example, some studies evaluated the channel's importance by the magnitude of weights [Hao Li, 2017], feature maps [Hu et al., 2016], or BN layers [Liu et al., 2017]. The similarity-based methods consider the approximate component as redundancy and a collection of clustering-based pruning methods [Lei et al., 2017] [Ding et al., 2019] have been developed. These studies have confirmed the effectiveness of both criteria for guaranteeing the model pruning performance. However, most of the existing studies adopt them independently and their integration may potentially lead to the improvement of pruning performance. Wang et al. [Wang et al., 2021] exploited the graph models based on similarity to evaluate layer redundancy but used the magnitude criterion to prune channels. Li et al. [Li et al., 2020] adopted two pruning processes that are carried out successively based on the variance and similarity of feature maps.

Inspired by the previous work, this paper attempts to construct a graph model at each layer and perform channel-wise pruning via the Graph Convolutional Network (GCN). The idea behind this method is that considering the channels as nodes, a graph model can be used to express the characteristic of channels through node features and the similarity between channels through connections of edges. Additionally, GCNs can aggregate the information from one node and its neighbors, i.e. combining the magnitude and similarity criteria for comprehensive channel evaluation.

To build the graph model, the distribution of Global Average Pooling (GAP) is used as the node feature, and the edge connections between nodes can be further determined based on the KL-divergence. Based on the graph, reinforcement learning (RL) is used to train an agent containing GCN and obtain the pruning scheme. The agent can repeat the "try-and-learn" process guided by the proposed objective function for multiple iterations to achieve the balance between pruned parameters and accuracy reduction. The benefit of the proposed pruning solution is confirmed through experiments based on the benchmark datasets for comparison with a range of models. The main technical contributions made in this work are summarized as follows.

(1) Adaptability to multiple models and layers. GAP ignores the position information of the feature map, which ensures the format invariance of our method in convolutional layers

or full-link layers. Also, the flexibility of GCN's input dimensions allows the proposed method to prune multiple layers simultaneously so that we can deal with residual connections conveniently.
(2) Automatically learning of criteria to determine sparsity and pruned channels. Our method is guided by an objective function where a balance ratio is selected to obtain the trade-off between the overall compression rate and performance reduction. For each layer, GCN can learn the optimal criterion under the supervision of RL without any additional assigned parameters.
(3) Outstanding model compression capability. Our proposed method can achieve better performance on multiple datasets and networks compared with the previous magnitude-based, similarity-based, and other SOTA algorithms.

2 Related Work

Magnitude-based pruning methods aim to remove the components with smaller magnitude in the network. The norm of convolution kernel is the most commonly used indicator, e.g., L1norm [Hao Li, 2017] and L2 norm [He et al., 2018] [Lee et al., 2021]. Existing studies also used BN layers or feature maps to evaluate channel importance. For example, Liu et al. [Liu et al., 2017] and Zhao et al. [Zhao et al., 2019] adopted the scaling coefficient of the BN layer for evaluation. Hu et al.[Hu et al., 2016] and Li et al[Li et al., 2020] evaluated the channel importance by sparsity and standard deviation of feature maps respectively.

Similarity-based pruning methods explore the network redundancy from approximate parts. Lei et al.[Lei et al., 2017] adopted K-means to obtain the clustering centers, by which all the convolutional kernels can be quantitatively represented. However, He et al.[He et al., 2019] argued that kernels closer to the clustering center have greater redundancy. Ding et al.[Ding et al., 2019] performed controllable and differentiable clustering by their proposed C-SGD method. The kernels are guided by loss function to be similar or identical during the training process and each cluster can only maintain one kernel and remove the rest after training.

Redundancy evaluation determines the target sparsity for each layer in the pruning process. A set of studies exploited the calculation of sparsity, e.g., PCA [Suau et al., 2018], graph theory [Wang et al., 2021]. However, most studies [Lemaire et al., 2019] attach masks to the weight matrix and add corresponding regularization terms to the loss function, hoping some weights can be compressed to zero automatically. The sparsity and pruned channels can also be determined simultaneously by the agent, but additional optimizing methods, e.g., RL [He et al., 2018], GAN[Lin et al., 2019] and gradient [Wu et al., 2018], are required for agent training.

Graph Convolutional Network is an effective tool for graph structural data modeling that has been widely used for pattern recognition [Liu et al., 2020] and semi-supervised classification[Zhuang and Ma, 2018]. GCNs can be classified as time-domain GCN and spectral-domain GCN according to the convolution domain. The former is an extension of CNN that aggregates node information by directly summing up neighborhood features; and the latter [Bruna et al., 2013] performs the approximate convolution operations through Laplacian transformation and can be often formulated as a first-order approximation.

3 Pruning Method

3.1 Preliminaries

For a CNN with N layers, let c_i denote the channel number of ith layer. Then the convolution kernel with size $w \times h$ can be represented as $f_i \in \mathrm{R}^{c_i \times c_{i-1} \times w \times h}$, and the feature map with size $W_i \times H_i$ can be denoted as $X_i \in \mathrm{R}^{c_i \times W_i \times H_i}$. Regarding fully connected layers in CNN as convolutional layers with 1×1 kernel, the inference process can be expressed as

$$X_i = \sigma(f_i * X_{i-1}) \tag{1}$$

where $*$ is the convolution operation, $\sigma(\cdot)$ is the nonlinear activation function, generally $Relu(\cdot)$, BN layers are omitted here for simplicity. Let $F = \{f_i\}$ denote the set of all parameters, I and l denote the input image and output label respectively, a CNN can be abbreviated as $l = \Phi_F(I)$.

Conducting channel-wise pruning for the CNN is equivalent to selecting channel indicator vector $m_i \in \{0,1\}^{c_i}$ for each layer and replacing the inference process as

$$X_i = m_i \odot \sigma(f_i * X_{i-1}) \tag{2}$$

where $\odot$ represents the Hadamard product on the channel dimension. Let $M = \{m_i\}$ represent the set of m_i. The pruned CNN can be written as $l = \Phi_{F,M}(I)$.

The GCN is defined on the graph structure. A graph $G = < V, E >$ is composed of a node-set $V = \{v_j\}$ with capacity α and an edge set $E = \{e = (v_{j_1}, v_{j_2}) | v_{j_1}, v_{j_2} \in V\}$. For a GCN with B layers, denoting the c_i-dimension node features of ith layer as $T_i \in R^{\alpha \times c_i}$, the inference process can be expressed as

$$T_i = \sigma(\hat{D}^{-\frac{1}{2}} \hat{A} \hat{D}^{-\frac{1}{2}} T_{i-1} k_{i-1}) \tag{3}$$

where $\hat{A}$ is adjacency matrix with self-ring, $\hat{D}$ is degree matrix of $\hat{A}$, $k_i \in R^{c_{i+1} \times c_i}$ is trainable parameters. Let $K = \{k_i\}$ represent all parameters, m and P denote output and input features, a GCN can be written as $m = \Psi_K(G, P)$.

Figure 1 overviews the proposed pruning framework.

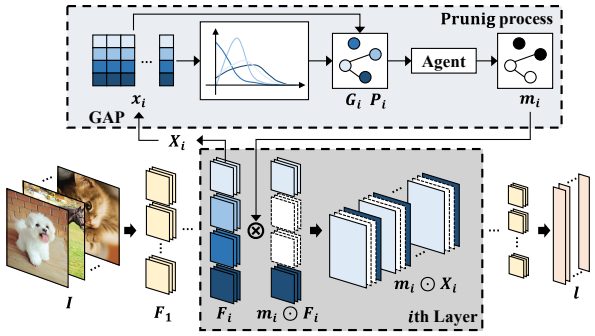

Figure 1: Pruning framework of our proposed method. For simplicity, only 4 channels are shown in ith layer. The prune scheme m_i is determined by the agent based on constructed graph G_i.

In our framework, the channel features P with each layer of CNN Φ are firstly exacted and a graph G is constructed, and then the pruning scheme m from a specialized GCN Ψ can be obtained.

3.2 Construction of Graph

Here, the construction method of G and the training method of Ψ is briefly introduced as follows. In ith layer, assembling all channels to a node-set V_i, the graph model can be constructed based on node (channel) features. Due to the capability of channel importance evaluation [Chen and Zhao, 2019] and format consistency among different layers, the distribution of GAP values is considered as the channel feature.

Let $\xi = \{I^k\}$ represent an image set with capacity n. When feeding kth sample I^k to the network, we get GAP value x_{ij}^k from ith layer and jth channel. As I^k traverses ξ, x_{ij}^k also forms a set $\chi_{ij} = \{x_{ij}^k\}$. To alleviate the difference between layers, χ_{ij} is further standardized to $\chi'_{ij} = \{x_{ij}'^k\}$ by $std(\chi_{ij})$. Since activation functions $Relu(\cdot)$ and the standardization process guarantee a bounded range for most of $x_{ij}^{k\prime}$, the data distribution of χ'_{ij} can be represented by an s-dimension vector p_{ij}. For the sampling interval $[0, u]$, the rth component of p_{ij} is

$$p_{ij}^{(r)} = \frac{1}{n}\left|\left\{x \middle| x \in \chi_{ij}, (r-1)\frac{u}{s} < x < r\frac{u}{s}\right\}\right| \quad (4)$$

Corresponding to V_i, the channel features for ith layer is defined as $P_i = \{p_{ij}\}$ that not only reflects the magnitude of feature maps but also ensures the consistency among similar channels. Figure 2 shows P_i of different networks and datasets.

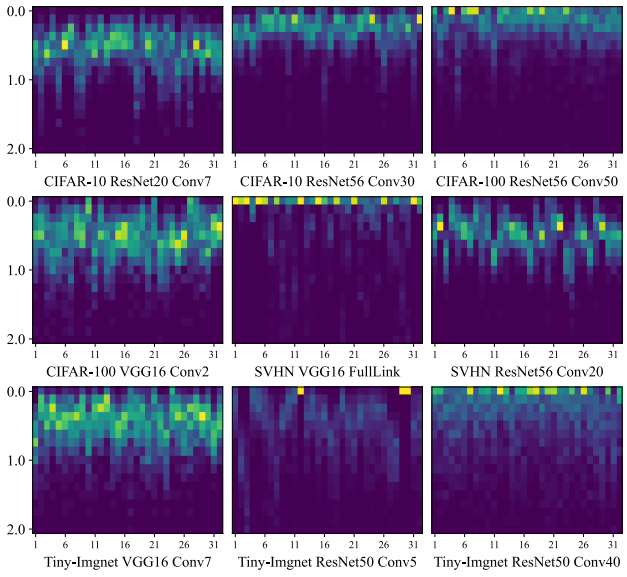

Figure 2: Channel features in various network layers and datasets. In each subgraph, the horizontal axis indicates the channel number and the vertical axis represents data distribution. For simplicity, 32 channels are selected to represent characteristics in each layer.

Edge set E_i can be determined by P_i. Since p_{ij} represents data distribution, we use KL divergence to measure channel distance. With a certain threshold θ, E_i can be written as $E_i = \{(v_{ij_1}, v_{ij_2}) | D_{KL}(p_{ij_1} | p_{ij_2}) < \theta, v_{j_1}, v_{j_2} \in V_i\}$. The selection of θ will be further discussed in Section 4.

3.3 Objective Function

The identification of an appropriate pruning scheme M for the pre-trained network Φ_F can be formulated as a minimum optimization problem. Similar to previous studies, the objective function is defined as

$$\Gamma(\Phi_F, M) = R_{prec} + \gamma R_{para} \quad (5)$$

where R_{prec} is the performance degradation after pruning, R_{para} is the proportion of pruned parameters, and γ represents the balance ratio. The FLOPs constraint is not selected here because its pruning effect is similar to that of R_{para}.

To avoid huge computational consumption, a batch of samples S is selected from the training dataset to evaluate performance degradation. Let $L(S, \Phi)$ denote the loss function, R_{prec} can be expressed as

$$R_{prec}(S, \Phi_F, M) = L^3(S, \Phi_{F,M}) / L^3(S, \Phi_F) \quad (6)$$

where the third power is used to enhance the sensitivity of model performance.

On the other hand, let $Para(\Phi)$ denote the number of non-zero parameters, R_{para} can be expressed as

$$R_{para}(\Phi_F, M) = Para(\Phi_{F,M}) / Para(\Phi_F) \quad (7)$$

With the correlation of layers, a greedy strategy is conducted to obtain M, in which we train GCN and prune for each layer sequentially. Therefore, the optimization objective for ith layer is

$$\min_{\Psi_i} \Gamma_i(m_i) = \Gamma(\Phi_F, M_{i-1} \cup m_i) \quad (8)$$

where Ψ_i is the abbreviated form of GCN Ψ_{K_i}, $M_{i-1} = \{m_1, m_2 \ldots m_{i-1}\}$ is the pruning scheme of the former $i-1$ layers.

3.4 Training Method for GCN

Given the optimization objective in 3.3, the training method of Ψ_i can be determined for ith layer. Since Γ is differentiable, taking $\bar{m}_i = \Psi_i(G_i, P_i)$ as pruning scheme, and optimize both Ψ_i and Φ by BP algorithm is a possible way. However, the previous studies (e.g., [Ding et al., 2021]) have revealed that continuous $\bar{m}_i$ can hardly converge to binary m_i even with the constraint term in loss function.

To avoid the converge problem, this work optimizes an agent containing GCN by RL (In detail, Policy Gradient) and obtain binary m_i from sampling. As shown in Figure 3, the environment of the agent is defined as G_i and P_i, which are extracted from a random batch of image ξ. The output of GCN $\bar{m}_i \in (0,1)^{c_i}$ represents the distribution of the agent's actions i.e. pruning schemes. Training of the agent is a repeated process of "try-and-learn". In each iteration, multiple m_i are sampled according to $\bar{m}_i$, and those who make Γ_i minimum is regarded as the label $\hat{m}_i$. With a loss function CrossEntropy, Ψ_i can be optimized by the BP algorithm.

To reduce the uncertainty caused by random selection of image batch ξ and ensure convergence, we also introduce a memory mechanism to the agent. The agent not only samples

m_i from current $\bar{m}_i$, but also takes the former τ iterations' $\hat{m}_i$ as possible attempts. The memory library of $\hat{m}_i$is denoted as Y in Figure 3.

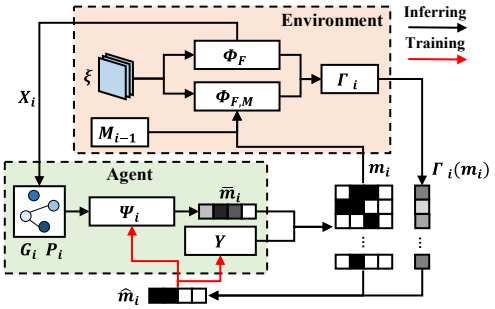

Figure 3: Training process of the agent in ith layer.

3.5 Pruning Method for ResNet

For networks containing residual connections, e.g., ResNet or DenseNet, the connections establish channel relevance between different layers, which brings considerable difficulties for model pruning in a layer-by-layer fashion.

To decouple the relevance, we divide all layers into different groups, as suggested in [Chen et al., 2021], so that layers in different groups are irrelevant and each group can be pruned separately. This requires no additional operations on the network while maintaining the potential pruning capability. The grouping scheme for the model pruning is illustrated in Figure 4.

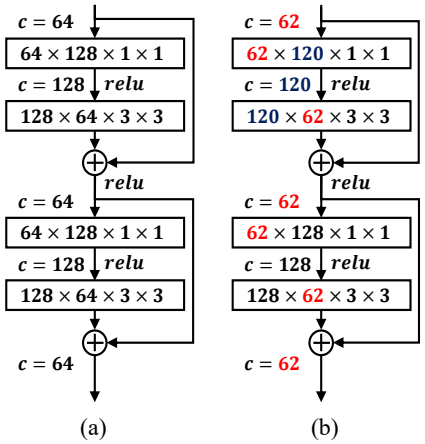

Figure 4: Grouping scheme and pruning results of ResNet. (a)The original network (b)The pruned network

In Figure 4, inner layers (blue) of the residual block are grouped separately, and layers containing residual connections (red) are grouped according to channel association. Once the pruned channels for the group with multiple layers are determined and their features are combined, the same process of pruning for individual layers can be performed similarly.

4 Experiment and Result

4.1 Datasets

The **CIFAR** [Krizhevsky and Hinton, 2009] is a widely used dataset consisting of CIFAR-10 and CIFAR-100, which have 10 and 100 classes respectively. Both of them contain 50000 training images and 10000 test images with a resolution of 32×32, and all the images are uniformly distributed across each class. The **Tiny-ImageNet** is a sub dataset of **ImageNet** [Deng et al., 2009], which contains 200 classes and 64×64 colored images. Each class has 500 training images, 50 validation images, and 50 test images.

During the model training, the images are randomly flipped horizontal and cropped with padding to avoid overfitting. The numbers of padding pixels for CIFAR and Tiny-ImageNet are set as is 4 and 8, respectively. In this work, no further effort on data augmentation is required.

4.2 Networks

The VGG [Simonyan and Zisserman, 2014] and ResNet [He et al., 2016] are the two popular benchmarks for model compression experiments and are adopted in this work. Here, the VGG-16 and ResNet-20, 50 and 56 are selected as the representative of the VGG family and the ResNet family. Due to the scale differences between datasets, models are adjusted accordingly to avoid excessive redundancy. For example, the channel number of full-link layers in VGG-16 is 4096 when training on Tiny-ImageNet, while that of CIFAR is 512.

4.3 Implementation Details

Models are well trained to their best performance before the layer-by-layer pruning is conducted. For each layer, we select $u = 3$, $s = 12$ to build a graph and design GCN Ψ with the structure $s - 16 - 32 - 16 - 1$. When training Ψ, 30 possible m will be evaluated for each iteration, where 20 are discretized from $\bar{m}$ and 10 are from prior iterations. After evaluation, $\hat{m}$ is regarded as the label for 5 times reinforcement training. Among all iterations, Ψ is trained with decreasing learning rate from 0.1 to 0.001 until convergence or reaching maximum iteration 300. Once the pruning scheme of the layer is determined by Ψ, corresponding channels will be pruned following a fine-tuning process of 30 epochs.

4.4 Performance Validation

Table 1 shows the performance of the proposed solution on a range of networks and datasets.

On the CIFAR-10 dataset, the pruning results show that even for small networks such as ResNet20, our method can achieve 0.7× compression of both FLOPs and parameters without accuracy reduction. Moreover, compared with the ResNet20 baseline, the pruned ResNet56 achieves higher accuracy with a smaller scale, which further indicates the effectiveness of our approach.

Figure 5 shows the pruning scheme of ResNet56 with different compression rates. It can be observed that the groups with residual connection (Res1-3) consisting of multiple lay-

ers are sensitive to pruning, and hence a few channels are removed even for 22% FLOPs compression. However, the layers in residual blocks (Conv2-54) are of low importance and hence are assigned with high sparsity to achieve overall network compression. The result of different layer sparsity shows that our method can efficiently carry out the automatic layer evaluation.

	Network	Accuracy (%)	FLOPs (M)	Parameters (M)
CIFAR-10	ResNet20 (92.25)	92.22	28.66 (69.53%)	0.20 (72.58%)
		91.58	20.38 (49.46%)	0.17 (61.49%)
	ResNet56 (93.72)	93.85	65.57 (51.69%)	0.56 (64.99%)
		92.75	28.89 (22.78%)	0.25 (29.50%)
	VGG16 (93.1)	93.27	134.22 (42.71%)	2.21 (14.50%)
		93.08	84.62 (26.93%)	1.06 (6.94%)
CIFAR-100	ResNet20 (68.38)	68.95	35.18 (85.36%)	0.25 (91.84%)
		66.14	16.54 (40.13%)	0.14 (50.58%)
	ResNet56 (72.86)	72.86	64.98 (51.23%)	0.52 (60.76%)
		72.22	60.60 (47.78%)	0.51 (60.17%)
	VGG16 (72.00)	72.00	136.47 (43.42%)	2.98(19.51%)
		71.64	99.82 (31.76%)	1.76(11.54%)
T-ImageNet	ResNet50 (59.66)	59.26	277.16 (82.52%)	14.63 (61.16%)
		57.67	117.98 (35.13%)	10.03 (41.92%)
	VGG16 (60.08)	59.04	791.79 (62.12%)	12.37 (35.95%)
		58.53	610.20 (47.87%)	10.07 (29.26%)

Table 1: Overall pruning results of our proposed approach on various networks and datasets. The baseline model accuracy is shown with the model name.

Being designed for the ImageNet dataset, VGG16 has amazing redundancy on CIFAR-10. Our proposed method can keep the accuracy almost unchanged with nearly 93% of parameters being removed. However, VGG's network structure is not as efficient as that of ResNet, so when compressed to a scale close to ResNet56 baseline, VGG16 cannot achieve similar accuracy.

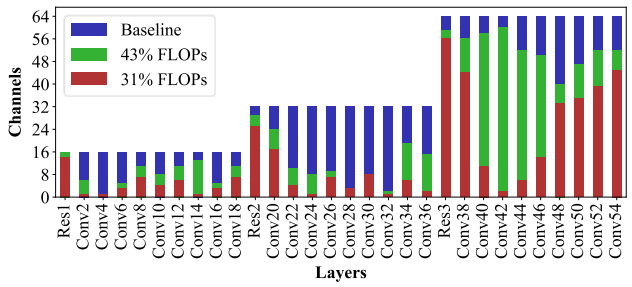

Figure 5: Pruning scheme of ResNet56 on CIFAR-10

On the CIFAR-100 dataset, the redundancy of each model declines in varying degrees. ResNet20 has a serious loss of accuracy when pruning 60% FLOPs. And ResNet56, while still able to retain accuracy with 50% FLOPs removal, is no longer able to accommodate greater compression. VGG16 has been least affected, our method still can retain accuracy by removing 88% parameters.

On the Tiny-ImageNet dataset, our method can maintain accuracy in slight pruning, but cannot conduct significant model compression. For example, we can compress ResNet50 with the removal of 39% parameters and only 0.4% accuracy reduction. But model degradation is inevitable when pruning over 64% FLOPs and 58% parameters.

4.5 Performance Comparison

The proposed solution is further extensively assessed through comparison with a range of existing solutions, including the magnitude-based methods (L1[Hao Li, 2017], SFP[He et al., 2018]), similarity-based methods (FPvG[He et al., 2019], QSFM[Wang et al., 2021]), agent-based methods (AMC [He et al., 2018], LFP [He et al., 2020]) and other SOTA methods (HRank [Lin et al., 2020], SCP [Kang and Han, 2020]) (the unnamed methods are denoted as the paper title abbreviations). In detail, L1 and SFP take L1norm and L2norm of filters as the importance index. The FPvG uses filter clustering methods to find out redundant channels, while QSFM evaluates similarity and prunes channel for each feature map pair. AMC and LFP train agents to automatically prune CNNs, where RL is also regarded as the training method. The HRank evaluates the channel importance by the rank of feature maps.

The pruning performance results of all algorithms on the VGG16 and ResNet56 on CIFAR-10 are shown in Table 2 and Table 3, respectively.

Method	Accuracy (%)	FLOPs (M)	Parameters (M)
QSFM	93.39→91.6 (-1.79)	235.08 (74.80%)	11.44 (75.00%)
SCP	93.69→93.23 (-0.46)	152.42 (48.50%)	7.86 (51.53%)
L1	93.25→93.4 (0.15)	206.00 (65.55%)	5.40 (35.41%)
FPE	93.03→92.49 (-0.54)	177.27 (56.41%)	3.30 (21.64%)
HRank	93.96→93.52 (-0.44)	145.61 (46.33%)	2.51 (16.46%)
Ours	**93.2→93.53 (0.33)**	**134.22 (42.71%)**	**2.21 (14.49%)**
HRank	93.96→91.23 (-2.73)	73.70 (23.45%)	1.78 (11.67%)
Ours	**93.2→93.08 (-0.12)**	**84.62 (26.93%)**	**1.06 (6.94%)**

Table 2: The comparison of different methods pruning VGG16 on CIFAR-10

Table 2 presents that our proposed approach achieves the lowest accuracy drop compared to other approaches in a wide range of compression ratios. When pruning VGG16 on CIFAR-10, QSFM, SCP, and FPE stop compressing with the removal of 30%-50% FLOPs, since the model begins to show unacceptable accuracy degradation. The HRank, L1 and our proposed solution demonstrate better performance and the proposed solution can even increase the model accuracy at certain pruning scenarios (reduction of around 60% FLOPs). Only a few methods attempt to significantly compress the model. The HRank and CUP can implement pruning schemes to remove 76.55% FLOPs. And Ours, although achieving a slightly less compression of FLOPs, have much better

performance on retaining the accuracy and compression of parameters.

Method	Accuracy (%)	FLOPs (M)	Parameters (M)
L1	93.04→93.06 (+0.02)	91.87 (72.43%)	0.73 (85.87%)
HRank	93.26→93.52 (+0.26)	89.66 (70.69%)	0.71 (83.51%)
QSFM	93.21→91.98 (-1.23)	73.56 (58.00%)	0.59 (69.10%)
Ours	**93.72→93.85 (+0.13)**	**65.56 (51.68%)**	**0.55 (64.27%)**
AMC	92.80→91.90 (-0.90)	63.42 (50.00%)	-
HRank	93.26→93.17 (-0.09)	63.38 (49.97%)	0.49 (56.97%)
SCP	93.69→93.23 (-0.46)	61.52 (48.50%)	0.44 (51.53%)
SFP	93.59→93.35 (-0.24)	60.12 (47.40%)	-
FPvG	93.59→92.93 (-0.66)	60.12 (47.40%)	-
LFP	93.59→93.34 (-0.25)	59.74 (47.10%)	-
Ours	**93.72→93.72 (+0.00)**	**58.29 (45.95%)**	**0.36 (42.07%)**
Ours	**93.72→92.75 (-0.97)**	**28.89 (22.77%)**	**0.25 (29.21%)**

Table 3: The comparison of different methods pruning ResNet56 on CIFAR-10

Table 3 shows that pruning ResNet56 on the CIFAR-10 dataset is a more challenging task. L1 and HRank can still increase model performance with light pruning, but for further compression, nearly all the methods suffer from accuracy reduction with the removal of 50% FLOPs. In comparison, our proposed solution can achieve the same accuracy as the baseline with only 45.95% FLOPs and 42.07% parameters, indicating a SOTA performance.

4.6 Ablation Study

The selection of threshold θ is an important factor for establishing the graph model. In this work, we select θ as 0.25 to achieve a sparse connection in each graph.

For validation, we further assign θ as 0 and 0.5 to represent the cases where the threshold is too small or too large. Experiments with VGG16 on CIFAR-100 show that inappropriate selection of θ leads to performance degradation. For example, with a certain compression ratio (57% FLOPs and 28% Paras) and selected thresholds above, only 71.4% and 71.3% accuracy can be maintained, which is significantly lower than the pruning result in Table 1(72% accuracy with 43% FLOPs and 20% Paras). That is because when θ is too small, our proposed method degenerates to a pure magnitude-based pruning method that cannot model a similar relationship between channels. In addition, when θ is too large, node features are homogenized due to excessive edge connections, which also degrades the performance.

The selection of γ balances the performance degradation and pruning ratio, which influences the pruned model size. Figure 6 shows when γ changes from 0(baseline) to 3000, the preserve ratio of multiple models on the CIFAR-10 dataset.

Figure 6 indicates that the size of the pruned model is not only affected by the selection of γ, but also by the redundancy of the model itself. For example, we can assign γ as 100 to compress 60% FLOPs of VGG16, but it is difficult to compress ResNet20. The objective function's sensitivity to model performance has automatically selected an appropriate compression ratio for each model. Therefore, to achieve a slight model compression, γ should be controlled within a small range, while for a large compression ratio, multiple rounds of pruning with suitable γ are needed.

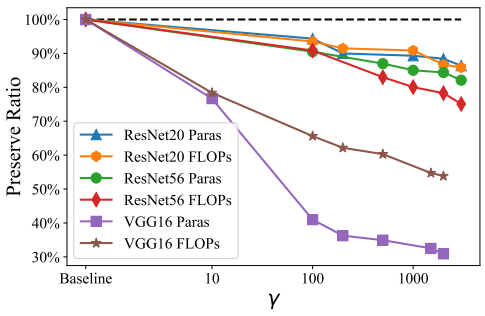

Figure 6: FLOPs and Parameters preserve the ratio of multiple models on CIFAR-10 as γ changes

5 Conclusion

This paper presented a novel GCN-based model pruning solution for efficient pruning of CNNs based on the outstanding performance of magnitude and similarity-based pruning methods. The proposed approach is implemented by combining the adjacent GAP distribution through GCN to identify the pruning scheme and RL is adopted as the training method. Extensive experiments based on VGG16 and ResNet demonstrated the effectiveness of our approach that can outperform the pure magnitude or similarity-based solutions and a set of other SOTA solutions. For future work, we will further validate the proposed method through deployment in different application domains.

References

[Hao Li, 2017] Asim Kadav Hao Li, Igor Durdanovic, Hanan Samet, Hans Peter Graf. Pruning Filters for Efficient ConvNets. In *International Conference on Learning Representations*, Toulon, France, April 24 - 26, 2017.

[Hu et al., 2016] Hengyuan Hu, Rui Peng, Yu-Wing Tai, et al. Network trimming: A data-driven neuron pruning approach towards efficient deep architectures. *arXiv preprint arXiv:1607.03250*, 2016.

[Liu et al., 2017] Zhuang Liu, Jianguo Li, Zhiqiang Shen, et al. Learning efficient convolutional networks through network slimming. In *International Conference on Computer Vision*, 2736-2744, 2017.

[Lei et al., 2017] Wang Lei, Huawei Chen and Yixuan Wu. Compressing Deep Convolutional Networks Using K-means Based on Weights Distribution. In *International Conference on Intelligent Information Processing*, Article 10, Bangkok, Thailand, 2017.

[Ding et al., 2019] Xiaohan Ding, Guiguang Ding, Yuchen Guo, et al. Centripetal SGD for Pruning Very Deep

Convolutional Networks With Complicated Structure. In *Conference on Computer Vision and Pattern Recognition*, 4938-4948, 15-20 June 2019.

[Wang et al., 2021] Zi Wang, Chengcheng Li and Xiangyang Wang. Convolutional neural network pruning with structural redundancy reduction. In *Conference on Computer Vision and Pattern Recognition*, 14913-14922, 2021.

[Li et al., 2020] Hang Li, Chen Ma, Wei Xu, et al. Feature statistics guided efficient filter pruning. *arXiv preprint arXiv:2005.12193*, 2020.

[He et al., 2018] Yang He, Guoliang Kang, Xuanyi Dong, et al. Soft Filter Pruning for Accelerating Deep Convolutional Neural Networks. In *International Joint Conference on Artificial Intelligence*, Stockholm, Sweden, July 13-19 2018.

[Lee et al., 2021] Jaeho Lee, Sejun Park, Sangwoo Mo, et al. Layer-adaptive Sparsity for the Magnitude-based Pruning. In *International Conference on Learning Representations*, Vienna, Austria, May 04 2021.

[Zhao et al., 2019] Chenglong Zhao, Bingbing Ni, Jian Zhang, et al. Variational Convolutional Neural Network Pruning. In *Conference on Computer Vision and Pattern Recognition*, 2775-2784, 15-20 June 2019.

[He et al., 2019] Yang He, Ping Liu, Ziwei Wang, et al. Filter Pruning via Geometric Median for Deep Convolutional Neural Networks Acceleration. In *Conference on Computer Vision and Pattern Recognition*, 4335-4344, 15-20 June 2019.

[Suau et al., 2018] Xavier Suau, Luca Zappella, Vinay Palakkode, et al. Principal filter analysis for guided network compression. *arXiv preprint arXiv:1807.10585*, 2018.

[Lemaire et al., 2019] Carl Lemaire, Andrew Achkar and Pierre-Marc Jodoin. Structured pruning of neural networks with budget-aware regularization. In *Conference on Computer Vision and Pattern Recognition*, 9108-9116, 2019.

[He et al., 2018] Yihui He, Ji Lin, Zhijian Liu, et al. Amc: Automl for model compression and acceleration on mobile devices. In *European Conference on Computer Vision*, 784-800, 2018.

[Lin et al., 2019] Shaohui Lin, Rongrong Ji, Chenqian Yan, et al. Towards optimal structured cnn pruning via generative adversarial learning. In *Conference on Computer Vision and Pattern Recognition*, 2790-2799, 2019.

[Wu et al., 2018] Zuxuan Wu, Tushar Nagarajan, Abhishek Kumar, et al. Blockdrop: Dynamic inference paths in residual networks. In *Conference on Computer Vision and Pattern Recognition*, 8817-8826, 2018.

[Liu et al., 2020] Yabo Liu, Yi Liu and Cheng Yang. Modulation recognition with graph convolutional network. *IEEE Wireless Communications Letters*, 9:624-627, 2020.

[Zhuang and Ma, 2018] Chenyi Zhuang and Qiang Ma. Dual graph convolutional networks for graph-based semi-supervised classification. In *Proceedings of the 2018 World Wide Web Conference*, 499-508, 2018.

[Bruna et al., 2013] Joan Bruna, Wojciech Zaremba, Arthur Szlam, et al. Spectral networks and locally connected networks on graphs. *arXiv preprint arXiv:1312.6203*, 2013.

[Chen and Zhao, 2019] S. Chen and Q. Zhao. Shallowing Deep Networks: Layer-Wise Pruning Based on Feature Representations. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 41:3048-3056, 2019.

[Ding et al., 2021] Xiaohan Ding, Tianxiang Hao, Jianchao Tan, et al. Lossless CNN Channel Pruning via Decoupling Remembering and Forgetting. In *International Conference on Computer Vision*, March 10, 2021.

[Chen et al., 2021] Tianyi Chen, Bo Ji, Tianyu Ding, et al. Only Train Once: A One-Shot Neural Network Training And Pruning Framework. *arXiv preprint arXiv:2107.07467*, 2021.

[Krizhevsky and Hinton, 2009] Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. *Handbook of Systemic Autoimmune Diseases*, 2009.

[Deng et al., 2009] Jia Deng, Wei Dong, Richard Socher, et al. Imagenet: A large-scale hierarchical image database. In *Conference on Computer Vision and Pattern Recognition*, 248-255, 2009.

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

[He et al., 2016] Kaiming He, Xiangyu Zhang, Shaoqing Ren, et al. Deep residual learning for image recognition. In *Conference on Computer Vision and Pattern Recognition*, 770-778, 2016.

[Wang et al., 2021] Zidu Wang, Xuexin Liu, Long Huang, et al. Model Pruning Based on Quantified Similarity of Feature Maps. *arXiv preprint arXiv:2105.06052*, 2021.

[He et al., 2020] Yang He, Yuhang Ding, Ping Liu, et al. Learning Filter Pruning Criteria for Deep Convolutional Neural Networks Acceleration. In *Conference on Computer Vision and Pattern Recognition*, 2006-2015, 13-19 June 2020.

[Lin et al., 2020] Mingbao Lin, Rongrong Ji, Yan Wang, et al. HRank: Filter Pruning Using High-Rank Feature Map. In *Conference on Computer Vision and Pattern Recognition*, 1526-1535, 13-19 June 2020.

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