CICC: Channel Pruning via the Concentration of Information and Contributions of Channels

Yihao Chen[1]
yihaochen@zju.edu.cn
Zhishan Li[2]
zhishanli@zju.edu.cn
Yingqing Yang[1]
yingqingyang@zju.edu.cn
Lei Xie[2]
leix@iipc.zju.edu.cn
Yong Liu[3]
yongliu@iipc.zju.edu.cn
Longhua Ma[4]
lhma_zju@zju.edu.cn
Shanqi Liu[2]
shanqiliu@zju.edu.cn
Guanzhong Tian*[1]
gztian@zju.edu.cn

[1] Ningbo Research Institute
Zhejiang University
Ningbo, China

[2] College of Control Science and Engineering
Zhejiang University
Hangzhou, China

[3] Zhejiang – Singapore Innovation and AI Joint Research Lab

[4] College of Information Science and Engineering
NingboTech University
Ningbo, China

Abstract

Channel pruning provides a promising prospect to compress and accelerate convolutional neural networks. However, existing pruning methods neglect the compression sensitivity of different layers and adjust the pruning rate through engineering tuning. To address this problem, we propose to assign the layer-wise pruning ratio via the concentration of information for the convolutional layers. Specifically, we introduce the rank and entropy of convolutional layers as indicators of the redundancy and amount of information, respectively. After that, we define a fusion function, which compromises these two indicators, to represent the concentration of information for the convolutional layers. Additionally, for pruning filters with interpretability and intuition, we propose to evaluate the importance of channels by leveraging Shapley values, which fairly distribute the *average marginal contributions* among them. Extensive experiments on various architectures and benchmarks demonstrate the promising performance of our proposed method (CICC). For example, CICC achieves an accuracy increase of 0.21% with FLOPs and parameters reductions of 45.5% and 40.3% on CIFAR-10. Besides, CICC obtains Top-1/Top-5 accuracy of 0.43%/0.11% with FLOPs and parameters reductions of 41.6% and 35.0% on ImageNet. It is worth noting that our method can still achieve excellent accuracy under high acceleration rates for pruning ResNet-110 on CIFAR-10.

* Corresponding author

1 Introduction

Convolutional Neural Networks (CNNs) have achieved excellent performance in computer vision tasks, such as image classification [9, 17, 43, 44], object detection [26, 28, 38, 39], semantic segmentation [2, 30, 40], *etc*. However, these models require a quantity of parameters and computational costs, which makes it difficult for deploying them on mobile and embedded devices. Even for efficient architectures (*e.g.*, residual connection [9] and inception module [44]), the over-parametrization and redundancy still exist and are always a challenge. Therefore, it is essential to reduce the memory footprint and computation overhead of the CNN-based models.

Network pruning is an effective way to accelerate and compress a model, and it can be mainly divided into two categories, *i.e.*, unstructured pruning [3, 4, 5, 6, 7, 8, 18] and structured pruning [21, 22, 35, 45, 46, 52]. Unstructured pruning directly deletes weight values in the layers to obtain sparsity. Nevertheless, the irregular sparse structure makes it difficult to leverage BLAS libraries [37]. On the contrary, structured pruning methods remove the entire filters or channels in the network, leaving a model with regular structures. In this paper, we focus on channel pruning while minimizing the accuracy drop of the model.

One open problem for channel pruning is how to assign an appropriate pruning rate for each layer. Recent works [12, 23, 33] tend to pre-define specific pruning rates for different layers empirically. However, this usually demands heuristic and engineering tuning [48]. Another critical problem is the selection of unimportant channels. Previous methods [10, 11, 19, 23] design hand-crafted pruning criteria to distinguish the importance of channels, but they do not leverage the interpretability of neural networks to evaluate the importance of channels.

To address these two open problems, we propose to assign the layer-wise pruning ratio via the concentration of information for the convolutional layers and prune the layers via the contributions of channels (CICC), as shown in Fig. 1. We first feed randomly sampled image batches to the pre-trained model to get the rank and entropy, which indicate the redundancy and amount of information [23, 32], respectively, for the outputs of convolutional layers. Then, we define a fusion function, which compromises these two indicators, to obtain an overall indicator as the concentration of information for the layers. After that, we assign the layer-wise pruning rate according to the fusion values. In the pruning phase, different from previous works [10, 19, 23, 32] that prune the channels by self-calculating the importance score, we bring up a theoretical basis: pruning the channels that contribute the least to loss optimization. Shapley values [42] are naturally fit for evaluating the contributions of channels, which could explicitly model the importance of the channels with feature attribution explanation by fairly distributing the *average marginal contributions* among them. When calculating the rank and entropy of the layers, we also compute the Shapley values. The channels with the lowest Shapley values represent they contribute less to the optimization, so pruning them leads to less harm to the performance of the model.

Contributions: To summarize, our contributions are as follows: (1) We define a fusion function which compromises the rank and entropy to obtain an overall indicator as the concentration of information for the convolutional layers. Then we assign the layer-wise pruning ratio based on the fusion values. (2) We tap into Shapley values as a powerful tool to evaluate the contributions of different channels and propose that contributions to the loss optimization should be a sound pruning criterion. (3) Extensive experiments for pruning backbones on the tasks of image classification on CIFAR-10 [16] and ImageNet [41], and object detection on COCO [25], demonstrate the excellent performance of our method.

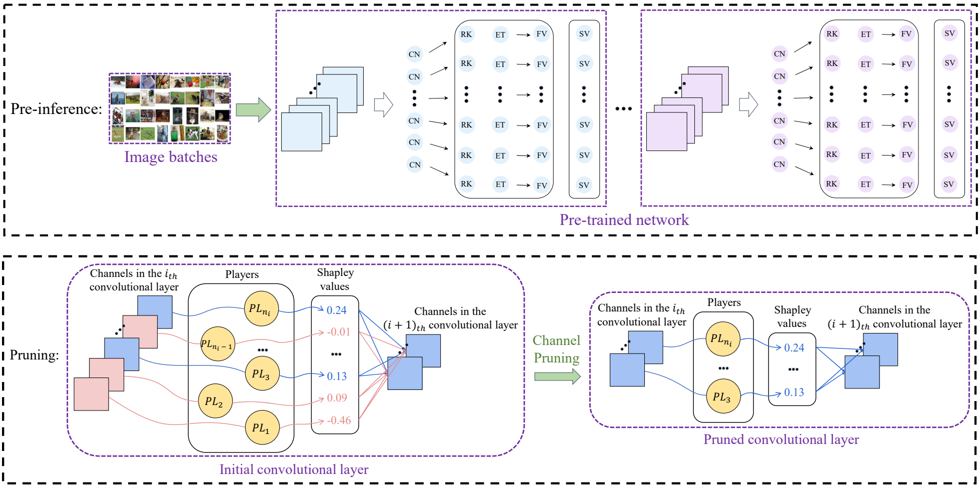


Figure 1: The framework of our method is divided into two phases. **(1)** Pre-inference: We feed randomly sampled image batches to obtain the rank and entropy (denoted by "RK" and "ET"), the corresponding fusion values (denoted by "FV") and the Shapley values (denoted by "SV") of each channel (denoted by "CN") in the convolutional layers. **(2)** Pruning: The channels in a layer are regarded as players (denoted by "PL"), and a negative Shapley value indicates that the player poses an adverse contribution to the cooperation. In each layer, the channels with the smallest Shapley values (pink squares) are discarded.

2 Related Work

Pruning criteria: Structured pruning methods prune a model in filter or channel levels. The discarded filters or channels reduce the model complexity and capacity but will inevitably harm the accuracy of the model [10]. Therefore, removing the least important filters or channels is an accepted way to minimize the decrease in accuracy. Prior works employ multiple criteria to approximate the importance of the filters to remove the unimportant ones, such as ℓ_1-norm [19], ℓ_2-norm [10], geometric median [11], rank [23] and statistics information computed from the next layer [33].

Pruning rate: Recent works pre-define specific pruning rates for different layers, which indicates that we know the percentage of filters/channels to be pruned in advance [48]. Early works [10, 11, 32, 50] adopt a constant pruning ratio to prune the same percentage of filters or channels in each convolutional layer. In contrast, HRank [23], ThiNet [33] and CP [12] set different pruning rates for each layer empirically. PFEC [19] and CC [20] prune fewer filters in the sensitive layers while pruning more aggressively in the insusceptible layers.

Pruning schedule: Pruning schedules to prune a network are generally categorized into three typical choices: (1) One-shot [19, 48]: Prune the filters of multiple convolutional layers at once. NISP [51], PFEC [19] and CC [20] prune the network by removing filters with the least importance in a single step and fine-tune to retain the performance. (2) Progressive [48]: Train and prune the network simultaneously. SFP [10] and FPGM [11] discard the least important filters at the end of each training epoch. (3) Iterative [19, 48]: Prune each layer and fine-tune the network, then repeat the process until the target sparsity is achieved. Previous studies [32, 33] prune and fine-tune the network layer by layer, and train the pruned model again when all the layers are pruned.

3 Methodology

3.1 Preliminaries

We assume that a CNN-based network has L convolutional layers, and n_i is the number of filters for the i_{th} convolutional layer $\mathcal{C}_i$. Let h_i and w_i be the height and weight of the feature maps in $\mathcal{C}_i$. $\mathcal{C}_i$ consists of a set of filters $F_i = \{F_i^1, F_i^2, \ldots, F_i^{n_i}\} \in \mathbb{R}^{n_i \times n_{i-1} \times k_i \times k_i}$, where k_i is the kernel size of the filters.

In channel pruning, n_i channels in $\mathcal{C}_i$ can be divided into two groups, *i.e.*, the removed ones $U_i = \{\mathcal{C}_i^{U_i^1}, \mathcal{C}_i^{U_i^2}, \ldots, \mathcal{C}_i^{U_i^{u_i}}\}$ and the remaining ones $Q_i = \{\mathcal{C}_i^{Q_i^1}, \mathcal{C}_i^{Q_i^2}, \ldots, \mathcal{C}_i^{Q_i^{q_i}}\}$, where u_i and q_i denote the number of removed and remaining channels, respectively.

Assume that $\psi(\mathcal{C}_i^j)$ represents the importance of the j_{th} channel in $\mathcal{C}_i$. Hence, channel pruning can be formulated as an optimization problem:

$$\min_{g_i^j} \sum_{i=1}^{L} \sum_{j=1}^{n_i} g_i^j \psi(\mathcal{C}_i^j), \quad s.t. \sum_{j=1}^{n_i} g_i^j = u_i, \tag{1}$$

where g_i^j is an indicator function which is 1 if $\mathcal{C}_i^j \in U_i$, or 0 if $\mathcal{C}_i^j \in Q_i$. Our objective is to minimize the information of the removed channels, *i.e.*, to identify the least important channels.

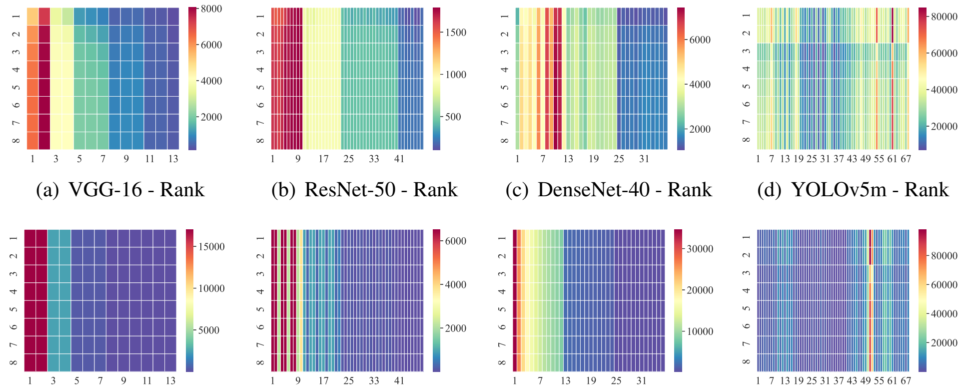
3.2 Concentration of Information

A convolutional layer with low rank represents that it contains a lot of *redundant information*, so it can be compressed into a more compact one. Previous work [23] illustrates the rank of each feature map under different image batches almost remains the same. Inspired by this, we find that a small number of image batches can estimate the rank of convolutional layers via their outputs, demonstrated in Fig. 2(a) ~ Fig. 2(c). Thus, we first sum the rank of feature maps in each layer by feeding B images randomly sampled from N ones. Then, we get the average rank per channel for the i_{th} convolutional layer $\mathcal{C}_i$:

$$R(\mathcal{C}_i) = \frac{\sum_{b=1}^{B} \sum_{j=1}^{n_i} Rank(F_i^j(b, j, :, :))}{n_i} \tag{2}$$

Entropy measures the disorder or uncertainty, *i.e.*, *the amount of information*, of a system [34]. In channel pruning, a convolutional layer with low entropy indicates that channels in it are less informative. Moreover, we find that the estimation of entropy for the convolutional layers is similar to that of rank, as shown in Fig. 2(e) ~ Fig. 2(g). Given B input images, we first map the tensor of the channel $\mathcal{C}_i^j$ between 0 and 1 with softmax function, so that the outputs of the channels in $\mathcal{C}_i$ can be regarded as the probability distribution:

$$p(\mathcal{C}_i^j) = Softmax(\mathcal{C}_i^j) = \frac{\sum_{b=1}^{B} e^{F_i^j(b,:,:,:)}}{\sum_{b=1}^{B} \sum_{j=1}^{n_i} e^{F_i^j(b,j,:,:)}}. \tag{3}$$



(a) VGG-16 - Rank (b) ResNet-50 - Rank (c) DenseNet-40 - Rank (d) YOLOv5m - Rank

(e) VGG-16 - Entropy (f) ResNet-50 - Entropy (g) DenseNet-40 - Entropy (h) YOLOv5m - Entropy

Figure 2: The average rank and entropy per channel for the outputs of convolutional layers under different batches of input images. The x-axis represents the indices of convolutional layers and the y-axis is the batches of images. The columns of subfigures demonstrate that the rank and entropy for the outputs of convolutional layers is almost unchanged, regardless of the image batches.

Next, the average entropy per channel of $\mathcal{C}_i$ is calculated as:

$$H(\mathcal{C}_i) = -\frac{\sum_{j=1}^{n_i} p(\mathcal{C}_i^j)\log p(\mathcal{C}_i^j)}{n_i} \tag{4}$$

Fig. 2 demonstrates that the rank and entropy for the outputs of convolutional layers is almost unchanged but with slight fluctuations under different image batches. Besides, the internal changes between the rank and entropy are not completely consistent. Thus, to eliminate the inconsistency and leverage the complementarity of information between these two indicators, we normalize them to the range $[a,b]$ and define a fusion function, which compromises these two indicators, to obtain an overall indicator as the concentration of information for the convolutional layers:

$$O(\mathcal{C}_i) = \prod_{Y,Z}((b-a)\frac{Y-\min Z}{\max Z - \min Z} + a), \tag{5}$$

where Y represents $R(\mathcal{C}_i)$ and $H(\mathcal{C}_i)$, and Z represents $\{R(\mathcal{C}_i)\}$ and $\{H(\mathcal{C}_i)\}$ $\forall i \in \{1,2,\dots,L\}$, respectively. Then we also normalize $O(\mathcal{C}_i)$ to $[a,b]$. The overall indicator combines the characterization of rank and entropy, so it represents the concentration of information. The convolutional layers with smaller fusion values indicate they are less informative, so u_i should be set to a larger value.

3.3 Channel Pruning via Shapley Value

Shapley value emerges from the context where the players participate in cooperation. They collectively obtain a reward which is intended to be fairly distributed to each player according to the individual contribution, and such a contribution is a Shapley value [42]. We extend it to the channel pruning scenario: Considering a convolutional layer in a CNN-based model as a game where individual channels in it "cooperate" to produce an output, we can attribute the layer-wise outcome to each channel.

Assume that a set $P = \{1, 2, \ldots, r\}$ consists of r players participating in a cooperation, and the subset $s \subseteq P$ denotes a coalition containing two or more players. We denote $v(s)$ as a characteristic equation defined on P if it satisfies $v(\varnothing) = 0$ and $\forall$ disjoint subsets $s_1, s_2 \subseteq P, v(s_1 \cup s_2) \geq v(s_1) + v(s_2)$.

The marginal contribution of the player t to all the coalitions containing t is calculated as:

$$\eta_t(v) = \sum_{s \in S_t} (v(s) - v(s \backslash \{t\})), \tag{6}$$

where S_t denotes the set that contains the player t from all subsets, and $v(s) - v(s \backslash \{t\})$ denotes the marginal contribution of the player t, *i.e.*, the contribution of the player t in coalition s.

Hence, the Shapley value for the player t is calculated as:

$$\varphi_t(v) = \sum_{s \in S_t} \frac{(|s|-1)!(r-|s|)!}{r!} (v(s) - v(s \backslash \{t\})) \propto \eta_t(v), \tag{7}$$

where $|s|$ represents the number of elements in the set s. Since $\varphi_t(v)$ is proportional to $\eta_t(v)$, it indicates the average marginal contribution of a player in the cooperation.

In the case of convolutional neural networks, we consider n_i channels $\{\mathcal{C}_i^1, \mathcal{C}_i^2, \ldots, \mathcal{C}_i^{n_i}\}$ in the convolutional layer $\mathcal{C}_i$ representing n_i players in the set $\mathcal{C}_i$. The function $\hat{f}$ maps each subset $m \subseteq \mathcal{C}_i$ of channels from activation outputs to real numbers for modeling the outcomes. The Shapley value of the channel $\mathcal{C}_i^j$ represents its average marginal contribution to the convolutional layer:

$$\varphi_{\mathcal{C}_i^j}(\hat{f}) = \sum_{m \in S_{\mathcal{C}_i^j}} \frac{(|m|-1)!(n_i-|m|)!}{n_i!} (\hat{f}(m) - \hat{f}(m \backslash \{\mathcal{C}_i^j\})). \tag{8}$$

Thus, we can reformulate Eqn.(1) as:

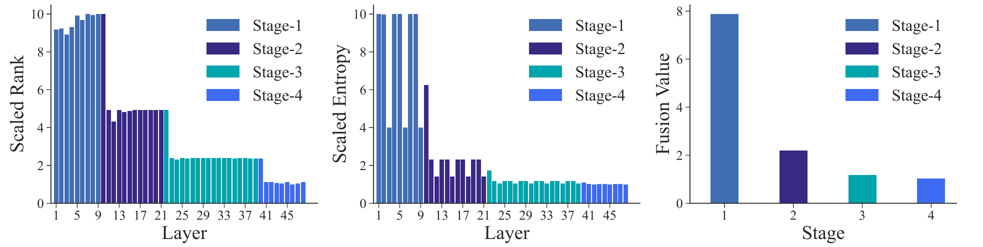
$$\min_{g_i^j} \sum_{i=1}^{L} \sum_{j=1}^{n_i} g_i^j \varphi_{\mathcal{C}_i^j}(\hat{f}), \quad s.t. \sum_{j=1}^{n_i} g_i^j = u_i. \tag{9}$$

4 Experiments

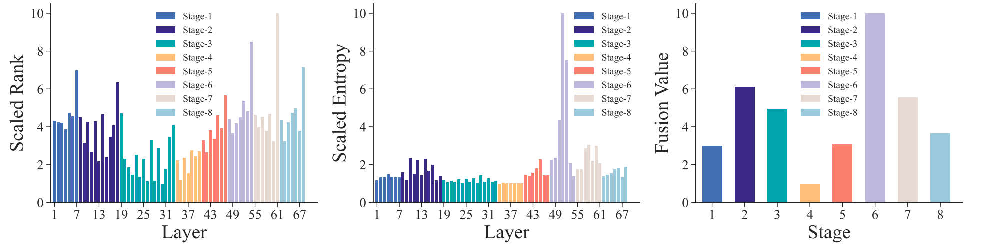
4.1 Experimental Settings

Benchmark datasets and models: To demonstrate the performance of our method, we conduct the experiments for pruning different architectures, including VGGNet, ResNet and DenseNet on CIFAR-10 and ImageNet, as well as YOLOv5 on COCO. We randomly sample 1024, 128 and 128 images to estimate the information of the convolutional layers for the backbones on CIFAR-10, ImageNet and COCO, respectively. The range is set $[1, 10]$ for scaling the rank, entropy and fusion values of the convolutional layers.

Configurations: On CIFAR-10 and ImageNet, we train the model with initial learning rate of 0.1 and batch size of 256 for 200 and 90 epochs, respectively. On COCO, we train the model with initial learning rate of 0.01 and batch size of 32 for 300 epochs. We use



(a) ResNet-50 - Rank (b) ResNet-50 - Entropy (c) ResNet-50 - Fusion Value



(d) YOLOv5m - Rank (e) YOLOv5m - Entropy (f) YOLOv5m - Fusion Value

Figure 3: The scaled average rank and entropy per channel for the outputs of convolutional layers and the corresponding fusion values of the stages. The layers in multiple colors indicate they are in different stages.

SGD as the optimizer. To evaluate the capabilities of models, we use Top-1 accuracy on CIFAR-10, Top-1 and Top-5 accuracies on ImageNet, and mAP on COCO. We adopt FLOPs and parameters reductions to evaluate the acceleration and compression ratios. On CIFAR-10, we iteratively prune and fine-tune the network for 20 epochs. On ImageNet and COCO, we employ the one-shot pruning schedule. The experiments are conducted on two NVIDIA RTX 3090 GPUs and two Tesla V100 GPUs.

4.2 Visualization for Concentration of Information

We first get the rank, entropy and the corresponding fusion values for the outputs of convolutional layers via Eqn.(5). Inspired by PFEC [19], we denote a stack of layers by a "stage" that keep the same feature map size. Inside each stage, we sum the fusion values and divide the result by the number of layers. As shown in Fig. 3, smaller fusion values indicate lower concentration of information. Thus, the pruning rates of the less important layers can be set to smaller values while the pruning rates of the more important layers can be set to larger ones. In our experiments, we prune the network in a per-stage fashion, *i.e.*, a pruning rate for a stage is used to prune the layers inside it.

4.3 Results and Analysis

4.3.1 Results on CIFAR-10

Tab. 1 shows the performance for pruned VGG-16, ResNet-56/110 and DenseNet-40 on CIFAR-10.

Model	Method	Base. Acc. (%)	Accl. Acc. (%)	Acc. ↓ (%)	FLOPs ↓ (%)	Params ↓ (%)
VGG-16	SSS [15]	93.96	93.02	0.94	41.6	**73.8**
	CP [12]	93.26	90.80	2.46	50.6	–
	CICC	93.91	93.17	**0.74**	**52.3**	45.7
	CICC	93.91	93.38	**0.53**	61.0	50.7
	HRank [23]	93.96	92.34	1.62	**65.3**	**82.1**
ResNet-56	GAL [24]	93.33	92.98	0.35	37.6	11.8
	ACTD [50]	93.69	93.76	-0.07	40.0	**50.0**
	CICC	93.39	93.60	**-0.21**	45.5	40.3
	AMC [13]	92.80	91.90	0.90	**50.0**	–
	FPGM [11]	93.59	93.26	0.33	52.6	–
	DBP [49]	93.69	93.27	0.42	52.0	40.0
	CICC	93.39	93.11	0.28	58.1	**43.9**
	Graph [51]	93.27	93.38	**-0.11**	**60.3**	43.0
ResNet-110	SFP [10]	93.68	93.86	-0.18	40.8	–
	HRank [23]	93.50	94.23	-0.73	41.2	39.4
	CICC	93.68	94.56	**-0.88**	**45.6**	**40.4**
	GAL [24]	93.50	92.74	0.76	48.5	**44.8**
	FPGM [11]	93.68	93.74	-0.16	52.3	–
	CICC	93.68	94.16	**-0.48**	**58.1**	44.0
DenseNet-40	CC [20]	94.81	94.67	**0.14**	**47.0**	51.9
	CICC	94.22	93.56	0.66	44.4	**60.8**
	HRank [23]	94.81	93.53	**1.28**	54.7	56.7
	CICC	94.22	92.54	1.68	**59.6**	**68.6**

Table 1: Comparison of pruned VGGNet, ResNet and DenseNet on CIFAR-10. "Base. Acc." and "Accl. Acc." refer to the accuracy of the baseline and pruned model. "Acc. ↓" is the accuracy drop between the pruned and baseline model. "FLOPs ↓" and "Params ↓" denote the FLOPs and parameters drop, respectively. The other tables follow the same convention.

VGG-16: Compared with SSS and CP, CICC achieves a lower accuracy drop (0.74% *v.s.* 0.94% by SSS and 2.46% by CP) and a larger FLOPs reduction (52.3% *v.s.* 41.6% by SSS and 50.6% by CP). Besides, CICC yields an acceleration ratio of 61.0% and compression ratio of 50.7%, obtaining a lower loss in accuracy (0.53%) than HRank (1.62%).

ResNet-56/110: For ResNet-56, CICC obtains an accuracy improvement better than ACTD (0.21% *v.s.* 0.07), while GAL and AMC harms the accuracy by 0.35% and 0.90%, respectively. Moreover, under larger acceleration ratio (58.1% *v.s.* 52.6% by FPGM and 52.0% by DBP) and compression ratio (43.9% *v.s.* 40.0% by DBP), CICC yields an accuracy drop of 0.28%, which is less than FPGM (0.33%) and DBP (0.42%), while Graph achieves an increase of accuracy by 0.11%.

For ResNet-110, CICC achieves an accuracy improvement of 0.88%, higher than SFP (0.18%) and HRank (0.73%). Additionally, CICC gains a higher accuracy increase (0.48%) than FPGM (0.16%), while GAL degrades the accuracy by 0.76%. With a slightly lower parameters reduction than GAL (44.0% *v.s.* 44.8%), CICC reduces more FLOPs than GAL and FPGM (58.1% *v.s.* 48.5% by GAL and 52.3% by FPGM).

DenseNet-40: Compared with HRank, CICC has the potential to compress the models with dense blocks. Specifically, though CC produces an accuracy drop of 0.14%, it only achieves a compression ratio of 51.9%. In contrast, 60.8% of parameters are removed by

CICC. Besides, CICC achieves a accuracy drop of 1.68%, producing a larger parameters reduction than HRank (68.6% *v.s.* 53.8%).

4.3.2 Results on ImageNet

Tab. 2 shows the performance for ResNet-50/101 on the large-scale ImageNet dataset.

Model	Method	Base. Top-1 Acc. (%)	Accl. Top-1 Acc. (%)	Base. Top-5 Acc. (%)	Accl. Top-5 Acc. (%)	Top-1 Acc. $\downarrow$ (%)	Top-5 Acc. $\downarrow$ (%)	FLOPs $\downarrow$ (%)	Params $\downarrow$ (%)
ResNet-50	DSA [36]	76.02	75.10	92.86	92.45	0.92	0.41	40.0	-
	CICC	76.13	75.70	92.86	92.75	**0.43**	**0.11**	41.6	35.0
	SFP [10]	76.15	74.61	92.87	92.06	1.54	0.81	41.8	–
	DECORE [1]	76.15	74.58	92.87	92.18	1.57	0.69	**44.7**	**42.3**
	DSA [36]	76.02	74.69	92.86	92.06	1.33	0.80	50.0	–
	TPP [47]	76.13	75.60	–	–	0.53	–	–	–
	CICC	76.13	75.29	92.86	92.47	0.84	**0.39**	**50.4**	**44.2**
	Fisher [27]	76.79	76.42	–	–	**0.37**	–	**50.4**	–
ResNet-101	FPGM [11]	77.37	77.32	93.56	93.56	0.05	0.00	42.2	–
	CICC	77.37	77.35	93.55	93.59	**0.02**	**-0.04**	**43.7**	**42.6**
	Rethinking [29]	77.37	75.27	–	–	2.10	–	47.0	–
	CICC	77.37	76.10	93.55	92.94	**1.27**	**0.61**	**54.4**	**54.0**

Table 2: Comparison of pruned ResNet on ImageNet.

For ResNet-50, CICC achieves 0.43%/0.11% Top-1/Top-5 accuracy drop, better than DSA (0.92%/0.41%), SFP (1.54%/0.81%) and DECORE (1.57%/0.69%). Moreover, CICC obtains 0.84%/0.39% Top-1/Top-5 accuracy drop, better than DSA (1.33%/0.80%) with a 44.2% parameters reduction. Besides, CICC achieves slightly higher Top-1 accuracy drop than TPP (0.53%) and Fisher (0.37%).

For ResNet-101, CICC achieves a negligible Top-1 accuracy drop of 0.02% and even a Top-5 accuracy improvement of 0.04%, which performs better than FPGM (0.05%/0.00% Top-1/Top-5 accuracy drop). Besides, under a FLOPs reduction of 54.4% and a parameters reduction of 54.0%, CICC obtains 1.27%/0.61% Top-1/Top-5 accuracy drop. In contrast, Rethinking degrades the Top-1 accuracy by 2.10%.

4.3.3 Results on COCO

Tab. 3 shows the performance for pruned YOLOv5s/m on COCO.

Model	Base. mAP (%)	Accl. mAP (%)	mAP $\downarrow$ (%)	FLOPs $\downarrow$ (%)	Params $\downarrow$ (%)
YOLOv5s	37.4	35.9	1.5	41.0	38.1
YOLOv5m	45.4	44.5	0.9	41.2	40.9

Table 3: Performance of pruned YOLOv5 on COCO with our proposed CICC.

For YOLOv5s, CICC achieves an mAP drop of 1.5% under 41.0% FLOPs and 38.1% parameters reductions. Besides, CICC obtains 0.9% mAP drop under 41.2% FLOPs and 40.9% parameters reductions for pruning YOLOv5m.

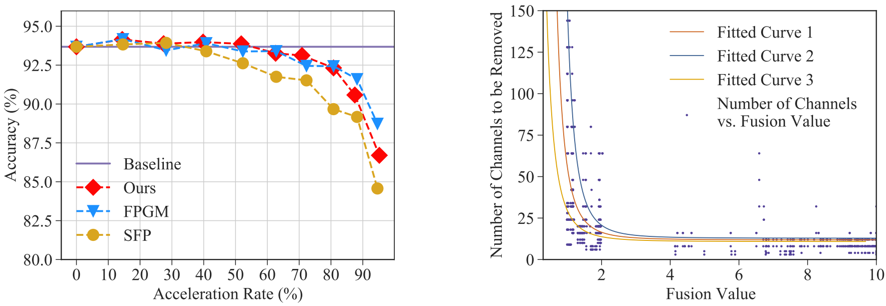
4.4 Ablation Studies

4.4.1 Varying Acceleration Rates

Fig. 4(a) shows the performance of our method compared with SFP [10] and FPGM [11] for pruning ResNet-110 on CIFAR-10 with one-shot schedule *w.r.t.* the acceleration rates. Our method achieves higher accuracy than the baseline model (93.68%) when the acceleration ratio is not more than 51.8%, while the performance of FPGM only exceeds the baseline model under 14.6% and 40.8% FLOPs reductions. This indicates that our method injects more effective sparsification into the model, which helps regularize the neural network and alleviate the over-fitting of an over-parameterized model [14].

4.4.2 Fitted Curve Between Pruning Ratio and Fusion Value

We empirically find a power function to fit the relationship between the number of channels to be removed *v.s.* the fusion value of the convolutional layer as: $f(x) = a \cdot x^b + c$, where $a = 44.58$, $b = -3.56$ and $c = 11.85$. Fig. 4(b) shows three curves for the estimation of the relationship, where Curve 2 and Curve 3 are shifted from Curve 1 derived from the function.



(a) Accuracy *w.r.t.* acceleration rates for ResNet-110 on CIFAR-10.

(b) The relationship of the number of channels to be removed *v.s.* the fusion value.

Figure 4: Results of ablation studies.

5 Conclusion and Future Work

In this paper, we define a fusion function, which compromises the rank and entropy, to represent the concentration of information for the convolutional layers. Based on the fusion values, we assign different pruning ratios for the layers. After that, we prune the layers by removing the least important channels evaluated by Shapley values. Extensive experiments on various backbones demonstrate the excellent performance of our method. In the future work, we will try to specify the quantitative relationships between the pruning ratio *v.s.* the fusion value of each convolutional layer in more scenarios and provide a general scheme for the selection of layer-wise pruning rate.

Acknowledgments

This work is supported in part by the Open Research Project of the State Key Laboratory of Industrial Control Technology, Zhejiang University, China (No. ICT2022B31), and in part by the National Natural Science Foundation of China under Grant U21A20484.

References

[1] Manoj Alwani, Yang Wang, and Vashisht Madhavan. Decore: Deep compression with reinforcement learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12349–12359, 2022.

[2] Vijay Badrinarayanan, Alex Kendall, and Roberto Cipolla. Segnet: A deep convolutional encoder-decoder architecture for image segmentation. *IEEE transactions on pattern analysis and machine intelligence*, 39(12):2481–2495, 2017.

[3] Miguel A Carreira-Perpinán and Yerlan Idelbayev. "learning-compression" algorithms for neural net pruning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8532–8541, 2018.

[4] Xin Dong, Shangyu Chen, and Sinno Pan. Learning to prune deep neural networks via layer-wise optimal brain surgeon. *Advances in Neural Information Processing Systems*, 30, 2017.

[5] Yiwen Guo, Anbang Yao, and Yurong Chen. Dynamic network surgery for efficient dnns. *Advances in neural information processing systems*, 29, 2016.

[6] Yiwen Guo, Anbang Yao, and Yurong Chen. Dynamic network surgery for efficient dnns. *Advances in neural information processing systems*, 29, 2016.

[7] Song Han, Huizi Mao, and William J Dally. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. *arXiv preprint arXiv:1510.00149*, 2015.

[8] Song Han, Jeff Pool, John Tran, and William Dally. Learning both weights and connections for efficient neural network. *Advances in neural information processing systems*, 28, 2015.

[9] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[10] Yang He, Guoliang Kang, Xuanyi Dong, Yanwei Fu, and Yi Yang. Soft filter pruning for accelerating deep convolutional neural networks. *arXiv preprint arXiv:1808.06866*, 2018.

[11] Yang He, Ping Liu, Ziwei Wang, Zhilan Hu, and Yi Yang. Filter pruning via geometric median for deep convolutional neural networks acceleration. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4340–4349, 2019.

[12] Yihui He, Xiangyu Zhang, and Jian Sun. Channel pruning for accelerating very deep neural networks. In *Proceedings of the IEEE international conference on computer vision*, pages 1389–1397, 2017.

[13] Yihui He, Ji Lin, Zhijian Liu, Hanrui Wang, Li-Jia Li, and Song Han. Amc: Automl for model compression and acceleration on mobile devices. In *Proceedings of the European conference on computer vision (ECCV)*, pages 784–800, 2018.

[14] Torsten Hoefler, Dan Alistarh, Tal Ben-Nun, Nikoli Dryden, and Alexandra Peste. Sparsity in deep learning: Pruning and growth for efficient inference and training in neural networks. *Journal of Machine Learning Research*, 22(241):1–124, 2021.

[15] Zehao Huang and Naiyan Wang. Data-driven sparse structure selection for deep neural networks. In *Proceedings of the European conference on computer vision (ECCV)*, pages 304–320, 2018.

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

[17] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25, 2012.

[18] Se Jung Kwon, Dongsoo Lee, Byeongwook Kim, Parichay Kapoor, Baeseong Park, and Gu-Yeon Wei. Structured compression by weight encryption for unstructured pruning and quantization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1909–1918, 2020.

[19] Hao Li, Asim Kadav, Igor Durdanovic, Hanan Samet, and Hans Peter Graf. Pruning filters for efficient convnets. *arXiv preprint arXiv:1608.08710*, 2016.

[20] Yuchao Li, Shaohui Lin, Jianzhuang Liu, Qixiang Ye, Mengdi Wang, Fei Chao, Fan Yang, Jincheng Ma, Qi Tian, and Rongrong Ji. Towards compact cnns via collaborative compression. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6438–6447, 2021.

[21] Zhishan Li, Yiran Sun, Guanzhong Tian, Lei Xie, Yong Liu, Hongye Su, and Yifan He. A compression pipeline for one-stage object detection model. *Journal of Real-Time Image Processing*, 18(6):1949–1962, 2021.

[22] Ji Lin, Yongming Rao, Jiwen Lu, and Jie Zhou. Runtime neural pruning. *Advances in neural information processing systems*, 30, 2017.

[23] Mingbao Lin, Rongrong Ji, Yan Wang, Yichen Zhang, Baochang Zhang, Yonghong Tian, and Ling Shao. Hrank: Filter pruning using high-rank feature map. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 1529–1538, 2020.

[24] Shaohui Lin, Rongrong Ji, Chenqian Yan, Baochang Zhang, Liujuan Cao, Qixiang Ye, Feiyue Huang, and David Doermann. Towards optimal structured cnn pruning via generative adversarial learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2790–2799, 2019.

[25] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer, 2014.

[26] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature pyramid networks for object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2117–2125, 2017.

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

[28] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C Berg. Ssd: Single shot multibox detector. In *European conference on computer vision*, pages 21–37. Springer, 2016.

[29] Zhuang Liu, Mingjie Sun, Tinghui Zhou, Gao Huang, and Trevor Darrell. Rethinking the value of network pruning. *arXiv preprint arXiv:1810.05270*, 2018.

[30] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3431–3440, 2015.

[31] Yao Lu, Wen Yang, Yunzhe Zhang, Jinhuan Wang, Shengbo Gong, Zhuangzhi Chen, Zuohui Chen, Qi Xuan, and Xiaoniu Yang. Graph modularity: Towards understanding the cross-layer transition of feature representations in deep neural networks. *arXiv preprint arXiv:2111.12485*, 2021.

[32] Jian-Hao Luo and Jianxin Wu. An entropy-based pruning method for cnn compression. *arXiv preprint arXiv:1706.05791*, 2017.

[33] Jian-Hao Luo, Jianxin Wu, and Weiyao Lin. Thinet: A filter level pruning method for deep neural network compression. In *Proceedings of the IEEE international conference on computer vision*, pages 5058–5066, 2017.

[34] David JC MacKay, David JC Mac Kay, et al. *Information theory, inference and learning algorithms*. Cambridge university press, 2003.

[35] 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.

[36] Xuefei Ning, Tianchen Zhao, Wenshuo Li, Peng Lei, Yu Wang, and Huazhong Yang. Dsa: More efficient budgeted pruning via differentiable sparsity allocation. In *European Conference on Computer Vision*, pages 592–607. Springer, 2020.

[37] Jongsoo Park, Sheng Li, Wei Wen, Ping Tak Peter Tang, Hai Li, Yiran Chen, and Pradeep Dubey. Faster cnns with direct sparse convolutions and guided pruning. *arXiv preprint arXiv:1608.01409*, 2016.

[38] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 779–788, 2016.

[39] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. *Advances in neural information processing systems*, 28, 2015.

[40] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer, 2015.

[41] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander Berg, and Li Fei-Fei. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, 115, 09 2014. doi: 10.1007/s11263-015-0816-y.

[42] L. S. Shapley. *17. A Value for n-Person Games*, pages 307–318. Princeton University Press, 2016. doi: doi:10.1515/9781400881970-018. URL https://doi.org/10.1515/9781400881970-018.

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

[44] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–9, 2015.

[45] Guanzhong Tian, Jun Chen, Xianfang Zeng, and Yong Liu. Pruning by training: a novel deep neural network compression framework for image processing. *IEEE Signal Processing Letters*, 28:344–348, 2021.

[46] Guanzhong Tian, Yiran Sun, Yuang Liu, Xianfang Zeng, Mengmeng Wang, Yong Liu, Jiangning Zhang, and Jun Chen. Adding before pruning: Sparse filter fusion for deep convolutional neural networks via auxiliary attention. *IEEE Transactions on Neural Networks and Learning Systems*, 2021.

[47] Huan Wang and Yun Fu. Trainability preserving neural structured pruning. *arXiv preprint arXiv:2207.12534*, 2022.

[48] Huan Wang, Can Qin, Yulun Zhang, and Yun Fu. Emerging paradigms of neural network pruning. *arXiv preprint arXiv:2103.06460*, 2021.

[49] Wenxiao Wang, Shuai Zhao, Minghao Chen, Jinming Hu, Deng Cai, and Haifeng Liu. Dbp: discrimination based block-level pruning for deep model acceleration. *arXiv preprint arXiv:1912.10178*, 2019.

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

[51] Ruichi Yu, Ang Li, Chun-Fu Chen, Jui-Hsin Lai, Vlad I Morariu, Xintong Han, Mingfei Gao, Ching-Yung Lin, and Larry S Davis. Nisp: Pruning networks using neuron importance score propagation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 9194–9203, 2018.

[52] Chenglong Zhao, Bingbing Ni, Jian Zhang, Qiwei Zhao, Wenjun Zhang, and Qi Tian. Variational convolutional neural network pruning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2780–2789, 2019.