Hindawi
Scientific Programming
Volume 2023, Article ID 9983781, 9 pages
https://doi.org/10.1155/2023/9983781

Research Article

ScoringNet: A Neural Network Based Pruning Criteria for Structured Pruning

Shuang Wang and Zhaogong Zhang

Heilongjiang University, Xuefu Road 73, Harbin 150080, China

Correspondence should be addressed to Zhaogong Zhang; 2013010@hlju.edu.cn

Received 30 November 2022; Revised 5 March 2023; Accepted 13 March 2023; Published 14 April 2023

Academic Editor: Dongpo Xu

Convolutional neural networks (CNNs) have shown their great power in multiple computer vision tasks. However, many recent works improve their performance by adding more layers and parameters, which lead to computational redundancy in many application scenarios, making it harder to implement on low-end devices. To solve this problem, model pruning methods are proposed, which aim to lower the computational and memory requirements of CNNs. In this paper, we propose ScoringNet, a neural network (NN) based pruning criteria for structured pruning procedure. ScoringNet generates a set of scores for each output channel in a model, which is used to reconstruct a pruned model later in a structured pruning way. ScoringNet can also use the gradient information to generate better scores, making the pruned model perform better. By using NNs, there are fewer hyperparameters, making it easier to implement. Experiment results demonstrate that the proposed ScoringNet can outperform or achieve competitive results compared to many state-of-the-art methods in both postpruning and pruning-at-initialization setups.

1. Introduction

Deep learning based computer vision solutions are using massive convolutional neural networks (CNNs) and large-scale datasets recently, which require heavy computational resources. However, computer vision-related tasks are usually deployed on low-end devices, where users' tolerance for latency and computational resources are limited. Therefore, compressing a large model to a smaller one without significant accuracy loss is important [1–13]. Network pruning is one of the methods which can lower the computational and memory requirements of CNNs. It generates a sparse network by removing redundant weights or activations through iterative training [2, 4, 5, 10, 14–16].

The early pruning methods focus on unstructured pruning, which removes single parameters from models [4, 5, 10, 14–16]. However, these methods require hardware and software support, making it hard to deploy in the real world situations. Therefore, many recent works are focusing on structured pruning [1, 2, 7]. Using structured pruning, a model can be pruned channel-wise, making the pruned model require no extra support than the original model.

The lottery ticket hypothesis (LTH) is introduced recently, which hypothesizes the existence of a sparse subnetwork in the original network, which can perform not worse than the original network if trained from scratch [17]. Inspired by the LTH, some variants are proposed, such as generalized LTH [18] and dual LTH [19]. However, most conventional training-based sparse pruning and LTH-inspired methods are using train-prune-retrain process, which is slow and expensive [20]. Therefore, pruning-at-initialization (PAI) is proposed to minimize the cost of pruning [20–23]. PAI prunes the models after initialization, so the training and pruning phase are decoupled, reducing the pruning process' complexity.

However, due to the design space of the pruning function, the handcrafted methods mentioned above, which are intuitive and explicit, are usually suboptimal solutions, and enumerating all functions with human labor under the space is impossible. Therefore, in Liu's work, evolution

strategy and genetic programming are used to obtain high-quality and transferable pruning functions. Therefore, in Liu's work [24], evolution strategy and genetic programming are used to obtain high-quality and transferable pruning functions. It shows that good pruning functions can be nonintuitive.

However, despite such nonintuitive functions being obtained using evolution strategy and genetic programming, it still needs a lot of computational resources. Since NNs are known for their ability to obtain implicit mappings from some features to other features, we use NNs to obtain the desired implicit mapping, which can approximate such nonintuitive functions mentioned above, with less time and computational resources consumed. To this end, we propose ScoringNet, a tiny NN to obtain the importance score of each channel, which is suitable for channel pruning, efficient in computational cost, and transferable, so that one can train ScoringNet with a small model, making the computational cost even less. Many studies have shown that the pruning ratio is better to set layer-wise instead of globally [1], so we use PRO [25] to obtain the desired layer-wise pruning ratio.

The input of ScoringNet is the statistical features of each channel, which has a fixed size. Former methods use norm-based and gradient-based scoring systems, which can be calculated by the statistical features we use in ScoringNet's input. So ScoringNet should at least outperform those methods as they can be derived from those statistical features.

Our main contributions are as follows:

(1) We propose ScoringNet, a neural network based pruning criteria that can obtain implicit mappings, which can approximate such nonintuitive functions, with less time and computational resources consumed. And it is the first NN-based pruning criteria to the best of our knowledge.

(2) We show that the ScoringNet we trained using a tiny model and a tiny dataset can also be used for pruning a large model.

(3) Experiments show that our model can prune NNs with relatively low precision loss, which works in both before training and after training settings, and its performance is competitive compared to many state-of-the-art algorithms.

2. Related Work

2.1. Pruning Methods

2.1.1. Unstructured Pruning. Unstructured pruning methods are simply pruning with a granularity of a single parameter [4, 5, 10, 14–16]. Although it is hard to speed up the inference procedure, it still has its value in research. Recent methods such as sparse structure selection (SSS) [26], single-shot network pruning based on connection sensitivity (SNIP) [21], gradient signal preservation (GraSP) [23], and evolving transferable pruning functions (ETPFs) [24] are using unstructured pruning manner. They make some breakthroughs by using feature maps, gradient flow and evolution strategy, and genetic programming, which can also be used for structured pruning.

2.1.2. Structured Pruning. Unstructured pruning methods focus on single parameters in a network, they do compress networks but are difficult to implement in current hardware and software settings. Therefore, many recent works are focusing on structured pruning, where network channels can be removed and the models can be practically compressed and accelerated without any further requirements [1, 2, 7]. A structured pruning method usually contains a reconstruction stage, which uses the output channels remaining and their weights to reconstruct pruned CNNs.

2.1.3. Lottery Ticket Theory. The Lottery Ticket Hypothesis (LTH): A randomly initialized, dense neural network contains a subnetwork that is initialized such that, when trained in isolation, it can match the test accuracy of the original network after training for at most the same number of iterations. This hypothesis suggests that one can prune a network's weight at initialization, which can not only reduce the redundant parameters, but also accelerate the training process thereafter. It is different from the original pruning methods, which prune the given trained dense networks, so lots of computational costs by training the dense networks are removed [17].

2.1.4. Pruning Ratio Optimization. Tuning pruning ratios is a complex task, as determining the level of redundancy in each layer can be challenging. Furthermore, retraining the model is often the only way to assess if the current pruning ratios are effective or not, which can be time-consuming and expensive. Thus, the pruning ratio optimizer (PRO) [25] is proposed to optimize pruning ratios in a greedy fashion based on the error in the final layer of the model.

2.2. Other Types of Acceleration Methods for CNNs. Here are some other common approaches that can accelerate the CNN models. Combined with pruning methods, these approaches can be used to achieve further acceleration.

2.2.1. Low-Rank Decomposition. Low-rank decomposition is an alternative for network compression [27–30]. It approximates convolutional operations by representing the weight matrix as a low-rank product of two smaller matrices. It reduces the computational costs in a different way from pruning, which means there are no convolutional operations left if it applies in a certain layer. Though low-rank decomposition can benefit the compression and speedup of CNNs, if the compression rate is high enough, the accuracy drop is not acceptable.

2.2.2. Quantization. Instead of using data type float or double, methods based on quantization aims to use integer weight to reduce the computational costs, therefore accelerating its inference [31, 32]. Recent quantization methods

have stepped further that they use low-bit integer weight to reduce the size of the model and computational cost even more, like ternary quantization [9] only uses 2 bit weight to represent (−1, 0, 1) which saves 16x in model size. However, lower bit often causes lower precision.

Despite quantization technique having all the benefits mentioned above, implementation of the methods based on quantization still requires both hardware and software support.

2.2.3. Knowledge Distillation. Knowledge distillation (KD) is another important branch of network compression techniques. By mimicking the output of a pretrained large model (also called teacher model), a relatively small model (also called student model) can obtain the knowledge from the teacher model, which makes the student model behave as good as its teacher [3, 33, 34]. The teacher model is usually treated as a black box in KD preset, which means only the input and output (softmax output in most cases) of the teacher model are accessible, but the parameters are not accessible.

3. Methods

The motivation for our work mainly comes from [24] that one can use combination features like $L_1 - Norm$ to represent a large number of weights in a single output channel. And we can extract the importance score of an output channel using the features mentioned above. Because neural networks (NNs) are known for their ability to obtain an implicit mapping from some features to other features, we can then use relatively small NNs to map those features to an important score. To this end, we designed ScoringNet using simple NNs to get an implicit mapping without causing heavy computational costs.

An illustration of the pruning procedure of ScoringNet is shown in Figure 1. There are three main components in our method, namely feature extractor, PRO, and ScoringNet. The feature extractor can transfer channels into their statistical features and can be set to global or layer-wise mode. We only use the pruning ratio of PRO since the ScoringNet cannot generate the pruning ratio itself. And the last part, ScoringNet, is designed to generate scores for every channel in the original model.

3.1. Notions. In this section, we define the notions. M denotes the original model that is to be pruned. D denotes the dataset used for pruning. K denotes the total number of layers. C_k represents the k-th output channel in a certain layer. L_n denotes the n-th layer in a certain model. w_i denotes the i-th weight in a certain output channel and g_i is its gradient. p_n denotes the pruning ratio of the n-th layer. f_n denotes the FLOPs in the n-th layer.

3.2. Feature Extractor. Feature extractor (FE) is designed to calculate the statistical features of all output channels. Motivated by [24], we also considered the statistical characteristics of other channels within the same layer as important factors because they give some extra information combined with the statistical characteristics of a certain output channel.

The statistical characteristics we used are listed below:

(i) Channel's l_1 norm

$$\sum_{w_i \in C_k} \text{abs}(w_i). \tag{1}$$

(ii) Channel's l_2 norm

$$\sqrt{\sum_{w_i \in C_k} w_i^2}. \tag{2}$$

(iii) Channel's mean

$$\frac{1}{\text{count}(C_k)} \sum_{w_i \in C_o} w_i. \tag{3}$$

(iv) Channel's variance

$$\frac{1}{\text{count}(C_k)} \sqrt{\sum_{w_i \in C_k} (\text{mean}(C_k) - w_i)^2}. \tag{4}$$

(v) Other channel's l_1 norm

$$\sum_{w_i \in L_n, w_i \notin C_k} \text{abs}(w_i). \tag{5}$$

(vi) Other channel's l_2 norm

$$\sqrt{\sum_{w_i \in L_n, w_i \notin C_k} w_i^2}. \tag{6}$$

(vii) Other channel's mean

$$\frac{1}{\text{count}(L_n)} \sum_{w_i \in L_n, w_i \notin C_k} w_i. \tag{7}$$

(vii) Other channel's variance

$$\frac{1}{\text{count}(L_n)} \sqrt{\sum_{w_i \in L_n, w_i \notin C_k} (\text{mean}(L_n) - w_i)^2}. \tag{8}$$

For gradient-related features, just replace w_i with its gradient g_i.

3.3. ScoringNet. It is hard to simply apply NNs to this kind of task because the input dimension is usually not a constant, and even if it were, the computational cost is unacceptable. To solve this input dimension problem, we use statistical characteristics of the weights instead of the weights themselves. Although we are utilizing those statistical characteristics, it may still not be enough to evaluate an output channel's importance, so gradient flow, which is considered to be an important factor in both pruning before training and pruning at the early phase of training tasks [21–23], together with the weights themselves, is also added into the input of our proposed ScoringNet.

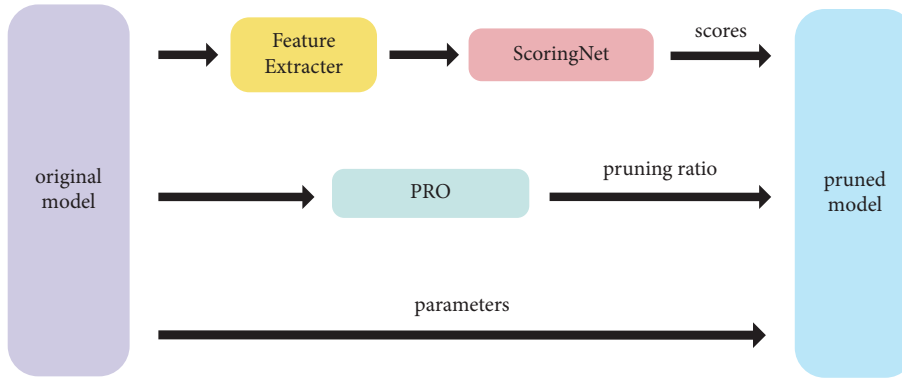

Figure 1: An illustration of the pruning procedure of ScoringNet. The original model's parameters are first fed into the feature extractor and PRO. With the statistical features, ScoringNet can generate scores for every channel. The pruned model then can be generated from the scores, pruning ratio, and the original parameters.

As shown in some previous works, only a single L1 or L2-norm based method can perform well enough, so simply applying a linear function to those statistical characteristics should also get an acceptable result, which should be not worse than only using one of them. NNs can be treated as a combination of linear functions and nonlinear functions (if activation functions like ReLU are used), which suit this kind of task well.

To this end, we propose weight-only networks, namely ScoringNet-s and ScoringNet-l, and gradient-related networks, ScoringNet-gs and ScoringNet-gl. "-s" stands for small, "-l" stands for large, and "-g" stands for gradient-related. The abovementioned small networks contain only 4 fully connected (FC) layers with 128 nodes in each layer, which is the same with all FC layers below, followed by 3 ReLU functions after the first three FC layers and the large ones contain 8 FC layers, with 7 ReLU functions after the first seven FC layers; for both networks, there is a normalization layer which normalizes all the "scores" to $(-5, 5)$. All ScoringNets mentioned above are used for the convolution layer. For linear layers, there are ScoringNet-s-l, ScoringNet-l-l, ScoringNet-gs-l, and ScoringNet-gl-l with the same settings mentioned above. Since the input vector only has 8 or 16 attributes, a very large NN is not necessary.

3.4. Training ScoringNet. Training such NNs is quite tricky since removing or masking out the pruned parameters is not a differentiable operation. One way to solve this problem is to imitate other methods' output masks, the other is to use a soft mask to keep the gradient flow.

For the soft mask solution, instead of reconstructing a new model, we multiply $\exp(\tanh(\text{score}(C_k)) - 1)$ to all weights in C_k, so that the backward propagation can work in our ScoringNet.

4. Experiment Result

4.1. Datasets. We evaluate our method and other pruning algorithms on CIFAR-10, CIFAR-100 [39], and ILSVRC2012 (ImageNet) [40]. These datasets are used for the single-label image classification task, which is usually considered to be a baseline among visual tasks. CIFAR-10 and CIFAR-100 have 10 and 100 classes, respectively, both consisting of 50,000 training images and 10,000 testing images. The 3-channel RGB images have 32×32 resolution, and the label distribution is equally balanced. At the training and evaluating stage, we use 40,000 randomly sampled training images to train the model, and the other 10,000 training set images form a validation set.

ImageNet is a large-scale benchmark compared to CIFAR-10 or CIFAR-100, consisting of a train-split of 1.28 million images and a val-split of 50,000 images. The 1,000 classes have balanced label distribution in both splits. The images have various resolutions, so the images are resized to 25×256 for batched training and testing. At the training and evaluating stage, 95% of the train-split (randomly sampled, referred as the training set in this section) are used to train the models, and perform validation on the remaining 5% train-split.

4.2. Base Models. We evaluate our method and other pruning algorithms on VGG-16/19 [44] and ResNet-50 [45]. VGG architecture features sequentially connected convolutional layers, followed by 3 fully connected layers for classification. Each convolutional or fully connected layer is rescaled by a batch normalization layer and activated by a ReLU function. VGG-16 and VGG-19 have 13 and 16 convolutional layers of 3×3 kernels, and the channel number range from 64 to 512.

4.3. Setups. For PAI setting, we first train a VGG-16 model for 5 epochs. Then, we train ScoringNets with the soft mask method on the half-trained VGG-16 on CIFAR-10, then use the trained ScoringNets on each model-dataset combination. We train convolution-related ScoringNet and linear-related ScoringNet separately. We use the output of the original model as ground truth, which means we do not want the pruned model to have better accuracy; we want the pruned model to have a similar behavior with the original model.

For experiments with CIFAR-10/100, we use the PAI setting; our method is compared with other SOTA-pruning methods with VGG-16/CIFAR-10 and VGG-19/CIFAR-

100. Base model and pruned model are trained for the same epochs and learning rate.

For experiments with ImageNet, we use a train-prune-retrain manner. In the retraining phase, the pruned models use the same data in the training phase. The ScoringNets, in this part, are trained with pretrained VGG-19 on CIFAR-100 and then transfered to prune pretrained VGG-16 and ResNet-50 with ImageNet.

4.4. Compared Methods. Our method is evaluated in comparison with several SOTA-pruning methods, including L1-norm [35], generative adversarial learning (GAL) [36], sparse structure selection (SSS) [26], generalized symmetric divergence (GSD) [37], high rank of feature maps (HRank) [38], slimming (SLIM) [41], evolving transferable pruning functions (ETPF) [24], single-shot network pruning based on connection sensitivity (SNIP) [21], gradient signal preservation (GraSP) [23], CP [2], runtime neural pruning (RNP) [42] and feature boosting and suppression (FBS) [43], channel pruning via Lookahead Search-Guided Reinforcement Learning (RL-MCTS) [46], and structural redundancy reduction (SRR) [47].

4.5. CIFAR-10. We conduct our experiment on dataset CIFAR-10 with base model VGG-16. To demonstrate the performance of our ScoreNets, we compare ScoreNets with L1, GAL, SSS, G-SD, and HRank.

As shown in Table 1, ScoringNet-gl gets the highest test accuracy, and all our results maintain the base accuracy with a smaller size and fewer FLOPs. Our models saved 51.0% to 52.1% FLOPs and 82.8% to 83.3% parameters, respectively, which is comparable with other SOTA methods. Deeper ScoringNets (ScoringNet-l and ScoringNet-gl) get better performance, and ScoringNets with extra gradient input (ScoringNet-gs and ScoringNet-gl) do not have a significant performance boost in this setup. Compared to the well-performed G-SD method, we are using small NNs instead of handcrafted metric, which cannot be derived from a shallow NN, showing that there should be NNs that can have a better performance.

4.6. CIFAR-100. On dataset CIFAR-100, we evaluate our methods with base model VGG-19. We choose SLIM, G-SD, ETPF, SNIP, and GraSP to compare with. In these methods, G-SD uses handcrafted metric, ETPF uses intuitive metric obtained by genetic programming, and SNIP and GraSP use gradient-based metrics.

As shown in Table 2, our models saved 66.7% FLOPs and 4.7% parameters, respectively. Compared to ETPF, our models get similar accuracy with slightly more FLOPs and parameters, but it is much quicker to get the desired pruning function with our approach. Compared to G-SD, our models saved more FLOPs and parameters and achieved similar accuracy drop.

4.7. ImageNet. On ImageNet, we evaluate our methods with base model VGG-16 and ResNet-50, and compare them with CP, G-SD-A, ETPF-A, RNP, SLIM, FBS, G-SD-B, ETPF-B, RL-MCTS, and SRR.

The experiment results with VGG-16 are shown in Table 3, ScoringNets outperform other methods with similar pruning rates. Since our ScoringNets used here are trained from VGG-19/CIFAR-100, the experiment results also show that our method is transferable through datasets and CNNs with plain architecture.

For experiment with ResNet-50, we add ScoringNet-gl-Scratch that train from ResNet-50/CIFAR-100 for comparison. Table 4 shows the experiment results with ResNet-50, and ScoringNets also get competitive results compared to other methods. Since ResNet has skip connections, which VGG-19 does not have, our results shows that we can transfer the ScoringNets trained from plain architecture to other architectures like ResNet. But the results also show that the ScoringNets trained from VGG-19 are not optimized, and one can get better ScoringNets by training them from the same architecture.

For ResNet-50, inspired by [48], we also designed a 90-epoch schedule, marked with "-90" in 4, which have better performance than the default 30-epoch schedule. These results show that it is possible to improve the pruned model using a better retraining schedule.

4.8. Training by Imitating Other Methods. There is another way to train ScoringNets, which is by imitating other methods' behavior. We use the L1-norm as a reference to train ScoringNets by taking the channel's L1-norm rank as ground truth.

We test both strategies on VGG-16/CIFAR-10, using ScoringNet-s and the target FLOPs to be pruned is 50%. As shown in Figure 2, ScoringNets trained in this way perform worse than the "train from scratch" strategy, but they do learn something useful from the L1-norm method. Based on this observation, we use the "train from scratch" manner to train our ScoringNets in other sections.

4.9. Implementation Details. We use Pytorch to implement the proposed ScoringNet approach. We use the stochastic gradient descent algorithm (SGD) as our optimizer with an initial learning rate of 0.01. Batch size, weight decay, and momentum are set to 128, 0.0005, and 0.9, respectively.

For experiments conducted on CIFAR-10/100, we use half-trained torchvision model as stated in 4.3. For experiments conducted on ImageNet, we use the pretrained torchvision model. After pruning, we retrain the pruned model for 30 epochs, with the learning rate being divided by 10 every 10 epochs.

For experiments marked with "−90", we use a different schedule, which is inspired by [48]. We use the stochastic gradient descent algorithm (SGD) as our optimizer with an initial learning rate of 0.1 for 10 epochs, then 0.01 for 15 epochs, then 0.001 for 25 epochs, and 0.0001 for 40 epochs.

5. Ablation Study

In this section, we discuss which factor is more important in our method. There are some main components in our method, the feature extractor (FE), pruning ratio optimization (PRO), and ScoringNet. Since ScoringNet depends on the output of FE, we only consider the impact of ScoringNet and PRO.

Table 1: Result on VGG-16/CIFAR-10.

Methods	Test Acc. (%)	Acc. ↓ (%)	FLOPs (M)	FLOPs ↓ (%)	Params (M)	Params ↓ (%)
L1 [35]	93.25 ⟶ 93.40	−0.15	211	34.2	5.40	64.0
GAL [36]	93.96 ⟶ 93.42	0.54	172	45.2	2.67	82.2
SSS [26]	93.96 ⟶ 93.02	0.94	183	49.6	3.93	73.8
G-SD [37]	93.45 ⟶ 93.68	−0.23	62	80.1	0.57	96.2
HRank [38]	93.96 ⟶ 93.43	0.53	**146**	**53.5**	**2.51**	**82.9**
ScoringNet-s (ours)	93.96 ⟶ 94.16	−0.20	165	51.5	2.50	83.0
ScoringNet-l (ours)	93.96 ⟶ 94.19	−0.23	166	51.7	2.48	83.1
ScoringNet-gs (ours)	93.96 ⟶ 94.11	−0.15	167	52.1	2.45	83.3
ScoringNet-gl (ours)	93.96 ⟶ 94.20	**−0.24**	164	51.0	2.52	82.8

The bold values represent the best values of the these methods.

Table 2: Result on VGG-19/CIFAR-100.

Methods	Test Acc. (%)	Acc. ↓ (%)	FLOPs (M)	FLOPs ↓ (%)	Params (M)	Params ↓ (%)
SLIM [41]	73.26 ⟶ 73.48	−0.22	256	37.1	5.0	75.1
G-SD [37]	73.40 ⟶ 73.67	−0.27	161	59.5	3.2	84.0
ETPF [24]	73.40 ⟶ 74.02	−0.62	**155**	**61.0**	2.9	85.5
SNIP [21]	74.16 ⟶ 72.84	1.32	—	—	**2.0**	**90.0**
GraSP [23]	74.16 ⟶ 71.95	2.21	—	—	**2.0**	**90.0**
ScoringNet-s	73.40 ⟶ 73.89	−0.49	162	60.1	3.0	85.1
ScoringNet-l	73.40 ⟶ 73.97	−0.57	162	60.3	3.0	85.2
ScoringNet-gs	73.40 ⟶ 73.95	−0.55	162	60.3	3.0	85.2
ScoringNet-gl	73.40 ⟶ 74.03	**−0.63**	162	60.2	3.0	85.2

The bold values represent the best values of the these methods.

Table 3: Result on VGG-16/ImageNet.

Methods	Top-1 Acc. (%)	Acc. ↓ (%)	Top-5 Acc. (%)	Acc. ↓ (%)	FLOPs ↓ (%)	Params ↓ (%)
L1 [35]	—	—	89.90 ⟶ 89.10	0.80	50.0	—
CP [2]	—	—	89.90 ⟶ 89.90	0.00	50.0	—
G-SD-A [37]	71.30 ⟶ 71.88	−0.58	90.10 ⟶ 90.66	−0.56	57.2	3.4
ETPF-A [24]	71.30 ⟶ 72.37	−1.07	90.10 ⟶ 91.05	−0.95	59.0	3.5
RNP [42]	—	—	89.90 ⟶ 86.67	3.23	66.7	—
SLIM [41]	—	—	89.90 ⟶ 88.53	1.37	66.7	—
FBS [43]	—	—	89.90 ⟶ 89.86	0.04	66.7	—
G-SD-B [37]	71.30 ⟶ 71.26	0.04	90.10 ⟶ 90.36	−0.26	69.7	5.2
ETPF-B [24]	71.30 ⟶ 71.64	−0.34	90.10 ⟶ 90.60	−0.50	66.9	4.8
ScoringNet-s	71.30 ⟶ 71.43	−0.13	90.10 ⟶ 90.34	−0.24	66.7	4.7
ScoringNet-l	71.30 ⟶ 71.59	−0.29	90.10 ⟶ 90.54	−0.44	66.7	4.7
ScoringNet-gs	71.30 ⟶ 71.63	−0.33	90.10 ⟶ 90.60	−0.50	66.7	4.7
ScoringNet-gl	71.30 ⟶ 72.73	−0.43	90.10 ⟶ 90.64	−0.54	66.7	4.7

Table 4: Result on ResNet-50/ImageNet.

Methods	Top-1 Acc.(%)	Acc. ↓ (%)	Top-5 Acc. (%)	Acc. ↓ (%)	FLOPs ↓ (%)	Params ↓ (%)
SSS [26]	76.12 ⟶ 74.18	1.94	92.86 ⟶ 91.91	0.95	31.0	—
G-SD-A [37]	76.15 ⟶ 76.21	−0.06	92.87 ⟶ 92.92	−0.05	29.1	14.1
G-SD-B [37]	76.15 ⟶ 75.85	0.30	92.87 ⟶ 92.66	0.21	44.3	23.2
RL-MCTS [46]	77.34 ⟶ 76.80	0.54	93.27 ⟶ 93.00	0.27	46.1	—
SRR [47]	76.13 ⟶ 75.76	0.37	92.86 ⟶ 92.67	0.19	44.1	—
ScoringNet-gs	76.13 ⟶ 75.52	0.61	92.86 ⟶ 92.44	0.42	45.5	23.7
ScoringNet-gl	76.13 ⟶ 75.54	0.59	92.86 ⟶ 92.45	0.41	45.4	23.7
ScoringNet-gl-Scratch	76.13 ⟶ 75.69	0.44	92.86 ⟶ 92.62	0.24	45.4	23.7
ScoringNet-gl-90	76.13 ⟶ 75.76	0.37	92.86 ⟶ 92.63	0.23	45.4	23.7
ScoringNet-gl-Scratch-90	76.13 ⟶ 75.83	0.32	92.86 ⟶ 92.65	0.21	45.4	23.7

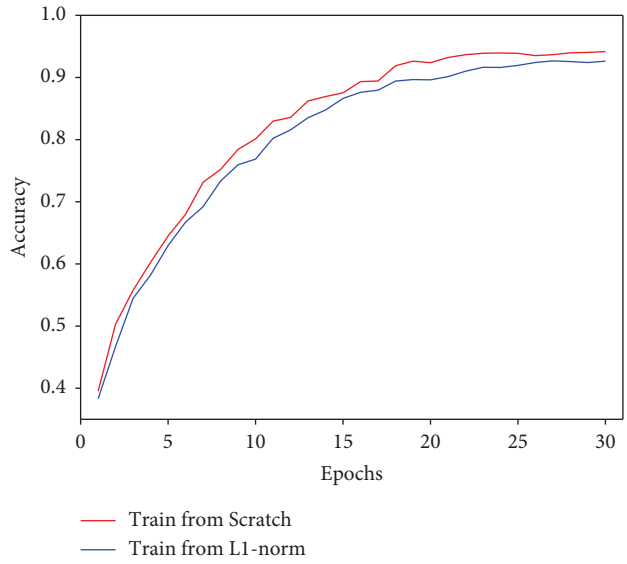

Figure 2: Experiment result of the training from scratch (top line) and training from L1-norm (bottom line). Both training processes are sharing the same setup (ScoringNet-s with target FLOPs to be pruned = 50.0%).

Table 5: Comparison between using global pruning ratio and layer-wise pruning ratio.

Methods	Test Acc. (%)	Acc. Lost (%)	FLOPs (M)	FLOPs ↓ (%)	Params (M)	Params ↓ (%)
ScoringNet-s(global)	93.96 ⟶ 92.86	1.10	167	51.0	2.50	83.0
ScoringNet-s(global)	93.96 ⟶ 92.53	1.43	118	65.3	1.86	87.3
ScoringNet-s(global)	93.96 ⟶ 90.14	3.82	84	75.4	1.36	90.7
ScoringNet-s(layer-wise)	93.96 ⟶ 94.16	−0.20	165	51.5	2.50	83.0
ScoringNet-s(layer-wise)	93.96 ⟶ 94.12	−0.16	117	65.5	1.83	87.5
ScoringNet-s(layer-wise)	93.96 ⟶ 94.05	−0.09	80	76.6	1.17	92.0

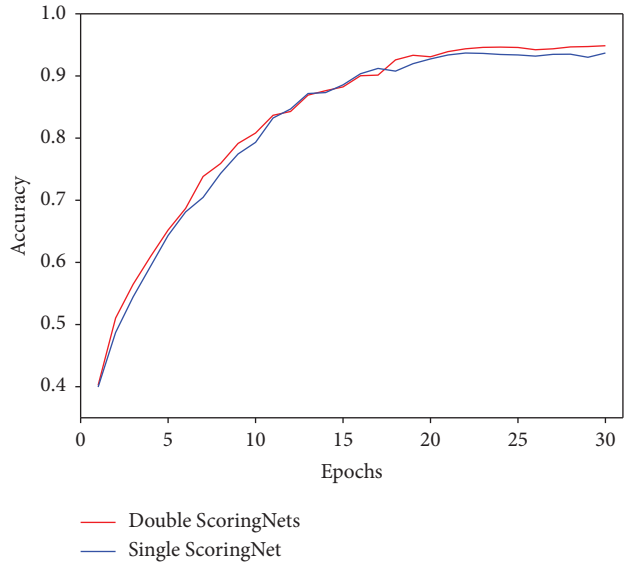

Figure 3: Experiment result of pruning by using double ScoringNets (top line) and pruning by using single ScoringNet (bottom line). Both training processes are sharing the same setup (ScoringNet-s with target FLOPs to be pruned = 50.0%).

5.1. Global Pruning Ratio vs. Layer-Wise Pruning Ratio. Without the layer-wise pruning ratio obtained from PRO, there is only a global pruning ratio that can be used. So we compare these two approaches with VGG-16/CIFAR-10. As shown in Table 5, the layer-wise pruning ratio outperforms the global pruning ratio by 1.3% to 3.91%, which proves that it is important to use a layer-wise pruning ratio. By increasing the pruning ratio, the global pruning ratio also suffers from performance degradation caused by information loss in several layers.

5.2. Single ScoringNets vs. Double ScoringNets. There are two ways to utilize ScoringNets to prune CNNs. The first one is to use a single ScoringNet to handle both convolutional layers, which are the main part of CNNs, and the fully connected layers at the end of CNNs. The second one is to use two ScoringNets to handle convolutional layers and fully connected layers separately. We compare these two approaches with VGG-16/CIFAR-10. Figure 3 demonstrates that the double ScoringNets approach performs better, which means despite the FE's output are statistical characteristics, there are notable differences in thestatistical features between convolutional layers and fully connected layers in neural networks. So, double ScoringNets approach is recommended.

6. Conclusion and Future Works

In this paper, we propose ScoringNet, a method to obtain pruning criteria for CNNs' structured pruning. We use the FE to extract the statistical characteristics so that the input dimension of ScoringNet can be fixed and the width of ScoringNet can be narrowed. We show that by utilizing gradient information, the quality of the pruned model can be improved. Because CNN models usually have a lot of layers, which have different information densities, we use a layer-wise pruning ratio which is generated by PRO. Considering the output of ScoringNet, the layer-wise pruning ratio can be obtained using other methods.

By proposing ScoringNet, we show that there is a way to use NNs to prune CNN models. If designed carefully, there should be better pruning criterion that uses NNs or other kinds of networks.

Data Availability

The data that support the findings of this study are available from the corresponding author upon reasonable request.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This work was supported by the Natural Science Foundation of China (61972135) and the Foundation of Graduate Innovative Research Project of Heilongjiang University (YJSCX2022-089HLJU).

References

[1] Y. Li, K. Adamczewski, L. Wen, S. Gu, T. Radu, and L. Van Gool, "Revisiting random channel pruning for neural network compression," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 191–201, New Orleans, LA, USA, June 2022.

[2] Y. He, X. Zhang, and J. Sun, "Channel pruning for accelerating very deep neural networks," in *Proceedings of the IEEE International Conference on Computer Vision*, pp. 1389–1397, Venice, Italy, October 2017.

[3] G. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," 2015, https://arxiv.org/abs/1503.02531.

[4] Y. Li, S. Gu, L. Van Gool, and T. Radu, "Learning filter basis for convolutional neural network compression," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 5623–5632, Seoul, Korea, June 2019.

[5] Z. Liu, M. Sun, T. Zhou, G. Huang, and T. Darrell, "Rethinking the value of network pruning," 2018, https://arxiv.org/abs/1810.05270.

[6] K. Wang, Z. Liu, Y. Lin, J. Lin, and S. Han, "Haq: hardware-aware automated quantization with mixed precision," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 8612–8620, Long Beach, CA, USA, June 2019.

[7] J. Ye, X. Lu, Z. Lin, and J. Z. Wang, "Rethinking the smaller-norm-less-informative assumption in channel pruning of convolution layers," 2018, https://arxiv.org/abs/1802.00124.

[8] X. Zhang, J. Zou, K. He, and J. Sun, "Accelerating very deep convolutional networks for classification and detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 38, no. 10, pp. 1943–1955, 2016.

[9] C. Zhu, S. Han, H. Mao, and W. J. Dally, "Trained ternary quantization," 2016, https://arxiv.org/abs/1612.01064.

[10] X. Dong, S. Chen, and S. Pan, "Learning to prune deep neural networks via layer-wise optimal brain surgeon," *Advances in Neural Information Processing Systems*, vol. 30, 2017.

[11] C. Li, Z. Wang, and H. Qi, "An efficient pipeline for pruning convolutional neural networks," in *Proceedings of the 2020 19th IEEE International Conference on Machine Learning and Applications (ICMLA)*, Miami, FL, USA, December 2020.

[12] H. Wang, C. Qin, Y. Zhang, and Y. Fu, "Neural pruning via growing regularization," in *Proceedings of the International Conference on Learning Representations*, Vienna, Austria, May 2021.

[13] S. Gao, F. Huang, W. Cai, and H. Huang, "Network pruning via performance maximization," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, Nashville, TN, USA, June 2021.

[14] P. Molchanov, T. Stephen, T. Karras, T. Aila, and J. Kautz, "Pruning convolutional neural networks for resource efficient inference," 2016, https://arxiv.org/abs/1611.06440.

[15] H. Yang, P. Liu, Z. Wang, Z. Hu, and Y. 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*, pp. 4340–4349, Long Beach, CA, USA, June 2019.

[16] Y. LeCun, J. Denker, and S. Solla, "Optimal brain damage," *Advances in Neural Information Processing Systems*, vol. 2, 1989.

[17] J. Frankle and M. Carbin, "The lottery ticket hypothesis: finding sparse, trainable neural networks," 2018, https://arxiv.org/abs/1803.03635.

[18] I. Alabdulmohsin, L. Markeeva, D. Keysers, and I. Tolstikhin, "A generalized lottery ticket hypothesis," 2021, https://arxiv.org/abs/2107.06825.

[19] B. Yue, H. Wang, T. A. O. Zhiqiang, K. Li, and Y. Fu, "Dual lottery ticket hypothesis," in *Proceedings of the International Conference on Learning Representations*, Vancouver, BC, Canada, May 2022.

[20] Y. Cai, W. Hua, H. Chen, G. E. Suh, C. De, and Z. Zhang, "Structured pruning is all you need for pruning cnns at initialization," 2022, https://arxiv.org/abs/2203.02549.

[21] N. Lee, T. Ajanthan, and P. H. S. Torr, "Snip: single-shot network pruning based on connection sensitivity," 2018, https://arxiv.org/abs/1810.02340.

[22] H. Tanaka, D. Kunin, D. L. Yamins, and S. Ganguli, "Pruning neural networks without any data by iteratively conserving synaptic flow," *Advances in Neural Information Processing Systems*, vol. 33, pp. 6377–6389, 2020.

[23] C. Wang, G. Zhang, and R. Grosse, "Picking winning tickets before training by preserving gradient flow," 2020, https://arxiv.org/abs/2002.07376.

[24] Y. Liu, S.-Y. Kung, and D. Wentzlaff, "Evolving transferable neural pruning functions," in *Proceedings of the Genetic and Evolutionary Computation Conference*, pp. 385–394, New York, NY, USA, July 2022.

[25] K. Kamma, S. Inoue, and T. Wada, "Pruning ratio optimization with layer-wise pruning method for accelerating convolutional neural networks," *IEICE - Transactions on Info and Systems*, no. 1, pp. 161–169, 2022.

[26] Z. Huang and N. Wang, "Data-driven sparse structure selection for deep neural networks," in *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 304–320, Tel Aviv, Israel, October 2018.

[27] J. Xue, J. Li, and Y. Gong, "Restructuring of deep neural network acoustic models with singular value decomposition," in *Proceedings of the Interspeech*, pp. 2365–2369, Lyon, France, August 2013.

[28] J. Ye, L. Wang, G. Li et al., "Learning compact recurrent neural networks with block-term tensor decomposition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 9378–9387, Salt Lake City, UT, USA, June 2018.

[29] X. Yu, T. Liu, X. Wang, and D. Tao, "On compressing deep models by low rank and sparse decomposition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 7370–7379, Honolulu, HI, USA, July 2017.

[30] Y. Li, S. Lin, J. Liu et al., "Rongrong. Towards compact cnns via collaborative compression," in *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition*, Nashville, TN, USA, June 2021.

[31] S. Jung, C. Son, S. Lee et al., "Learning to quantize deep networks by optimizing quantization intervals with task loss," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 4350–4359, Long Beach, CA, USA, June 2019.

[32] K. Zhao, S. Huang, P. Pan et al., "Distribution adaptive int8 quantization for training cnns," in *Proceedings of the AAAI Conference on Artificial Intelligence*, pp. 3483–3491, Washington, DC, USA, February 2021.

[33] Z. Wang, "Zero-shot knowledge distillation from a decision-based blackbox model," in *Proceedings of the International Conference on Machine Learning*, New York, NY, USA, July 2021.

[34] C. Li, Z. Wang, and H. Qi, "Online knowledge distillation with history-aware teachers," in *Proceedings of the International Joint Conference on Neural Networks (IJCNN)*, Shenzhen, China, July 2022.

[35] H. Li, A. Kadav, I. Durdanovic, H. Samet, and H. P. Graf, "Pruning filters for efficient convents," 2016, https://arxiv.org/abs/1608.08710.

[36] S. Lin, R. Ji, C. Yan et al., "Towards optimal structured cnn pruning via generative adversarial learning," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 2790–2799, Long Beach, CA, USA, June 2019.

[37] Y. Liu, D. Wentzlaff, and S.-Y. Kung, "Rethinking class-discrimination based cnn channel pruning," 2020, https://arxiv.org/abs/2004.14492.

[38] M. Lin, R. Ji, Y. Wang et al., "Hrank: filter pruning using high-rank feature map," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 1529–1538, Seattle, WA, USA, July 2020.

[39] A. Krizhevsky and G. Hinton, *Learning Multiple Layers of Features from Tiny Images*, Technical report, University of Toranto, Ontario, Canada, 2009.

[40] D. Jia, W. Dong, R. Socher, L. Li, K. Li, and L. Fei-Fei, "Imagenet: a large-scale hierarchical image database," in *Proceedings of the 2009 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 248–255, Miami, FL, USA, June 2009.

[41] Z. Liu, J. Li, Z. Shen, G. Huang, S. Yan, and C. Zhang, "Learning efficient convolutional networks through network slimming," in *Proceedings of the IEEE International Conference on Computer Vision*, pp. 2736–2744, Venice, Italy, October 2017.

[42] J. Lin, Y. Rao, J. Lu, and J. Zhou, "Runtime neural pruning," *Advances in Neural Information Processing Systems*, vol. 30, 2017.

[43] X. Gao, Y. Zhao, Ł. Dudziak, R. Mullins, and C. Xu, "Dynamic channel pruning: feature boosting and suppression," 2018, https://arxiv.org/abs/1810.05331.

[44] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," 2014, https://arxiv.org/abs/1409.1556.

[45] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the 2016 Computer Vision and Pattern Recognition*, pp. 770–778, IEEE, Las Vegas, NV, USA, June 2016.

[46] Z. Wang and C. Li, "Channel pruning via lookahead search guided reinforcement learning," in *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, New Orleans, LA, USA, June 2022.

[47] Z. Wang, C. Li, and X. Wang, "Convolutional neural network pruning with structural redundancy reduction," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, Nashville, TN, USA, June 2021.

[48] H. Wang, C. Qin, B. Yue, and Y. Fu, "Why is the state of neural network pruning so confusing? on the fairness, comparison setup, and trainability in network pruning," 2023, https://arxiv.org/abs/2301.05219.