Zhang C, Wang HZ, Liu HW *et al.* Fine-tuning channel-pruned deep model via knowledge distillation. JOURNAL OF COMPUTER SCIENCE AND TECHNOLOGY 39(6): 1238-1247 Nov. 2024. DOI: 10.1007/s11390-023-2386-8

Fine-Tuning Channel-Pruned Deep Model via Knowledge Distillation

Chong Zhang[1] (张　翀), *Student Member, CCF*
Hong-Zhi Wang[1] (王宏志), *Senior Member, IEEE, Member, CCF*
Hong-Wei Liu[1,*] (刘宏伟), *Member, CCF*, and Yi-Lin Chen[2] (陈熠琳)

[1] *Faculty of Computing, Harbin Institute of Technology, Harbin 150001, China*
[2] *School of Astronautics, Harbin Institute of Technology, Harbin 150001, China*

E-mail: 19B903094@stu.hit.edu.cn; wangzh@hit.edu.cn; liuhw@hit.edu.cn; 120l011005@stu.hit.edu.cn

Received April 4, 2022; accepted November 7, 2023.

Abstract Deep convolutional neural networks with high performance are hard to be deployed in many real world applications, since the computing resources of edge devices such as smart phones or embedded GPU are limited. To alleviate this hardware limitation, the compression of deep neural networks from the model side becomes important. As one of the most popular methods in the spotlight, channel pruning of the deep convolutional model can effectively remove redundant convolutional channels from the CNN (convolutional neural network) without affecting the network's performance remarkably. Existing methods focus on pruning design, evaluating the importance of different convolutional filters in the CNN model. A fast and effective fine-tuning method to restore accuracy is urgently needed. In this paper, we propose a fine-tuning method KDFT (Knowledge Distillation Based Fine-Tuning), which improves the accuracy of fine-tuned models with almost negligible training overhead by introducing knowledge distillation. Extensive experimental results on benchmark datasets with representative CNN models show that up to 4.86% accuracy improvement and 79% time saving can be obtained.

Keywords model compression, deep learning, knowledge distillation, fine-tuning

1 Introduction

In the field of big data analysis, deep convolutional neural networks (DCNNs) are steadily pushing the envelope of a variety of tasks, such as image classification[1–6], object detection[7, 8], and semantic segmentation[9, 10]. Nowadays, CNN (convolutional neural network) models are mainly proposed by experts, and their network structures are designed to be concise to a great extent. However, to learn from a huge volume of data and achieve a remarkable performance, the network still needs to be deep, with a large number of parameters. Such large CNNs are both computationally expensive and memory demanding. They can only be trained and deployed with the help of GPUs or other sophisticated hardware. Therefore, an intuitive method to deploy DCNNs in resource limited applications is to compress DCNN models.

To reduce computation of CNN models with limited performance degradation, model compression techniques have emerged. Many studies were proposed from different perspectives, including connection pruning[11, 12], sparsity regularization[13], parameter quantization[14], low rank approximation[15], and structural channel pruning[16–19].

The idea of removing redundant components in a neural network came from LeCun *et al.*[20]. With the help of information-theoretic ideas, it is found that by removing unimportant weights, a network could be easier to train and achieve better generalization, with even better performance. Inspired by this work, several studies on sparsity regularization and connection pruning have emerged, which either obtains sparse weight matrices by regularization in the training phase or prunes the network weights directly. However, the models pruned by these methods are often

Regular Paper
This work was supported by the National Natural Science Foundation of China under Grant No. U1866602.
*Corresponding Author

non-structured neural connections, resulting in their limited applications on general-purpose hardware or BLAS[21] (Basic Linear Algebra Subprograms) libraries. Some work compresses the models by low rank approximation[22] or encoding[23].

In contrast, channel pruning methods do not have this limitation, making them more suitable for compressing CNNs, since they prune entire convolutional channels away. Usually, there are three phases for a channel pruning method. Firstly, an importance assessment mechanism on convolutional channels is proposed. This mechanism may only consider local statistical characteristics of convolutional channels, such as the average percentage of zeros (APoZ)[24] or the average rank of kernel matrices[16]. It could also assess the influence of a kernel on the model's overall performance[17]. Secondly, guided by the assessment mechanism and a proper pruning rate, unimportant kernels are removed. To keep a structural balance and maintain accuracy, the importance of kernels is often compared inside the same layer, and the balance of pruning rate for different layers is also considered[17]. The third phase is called fine-tuning[25]. During this phase, the pruned networks have to be trained for several epochs with the same training dataset to restore accuracy.

Knowledge distillation[26] is a promising way to obtain a small model that retains the accuracy of a large one, by transferring the complex knowledge learned by the large model to the small one.

Among all the methods above, channel pruning methods are the most promising way to apply DCNN models on resource limited edge devices. The accuracy of the pruned model after fine-tuning may be close to or even exceed the accuracy of the original model.

But the increase of compression rate for channel pruning methods is urgent, and the accuracy drop after extreme pruning is a dominant bottleneck for modern channel pruning methods. On the other hand, to get an ideal fine-tuning accuracy, sometimes it takes a lot of training epochs, even more than training the original model. When the pruning rate is too large, the accuracy drops significantly and could not be improved by simply adding fine-tuning epochs. We notice that the deep convolutional models before and after channel pruning make a good pair of the teacher and the student. Inspired by this observation, we propose the Knowledge Distillation Based Fine-Tuning (KDFT) for channel-pruned deep models, a framework which utilizes knowledge distillation in the fine-tuning phase for a channel pruning method to better and more efficiently restore accuracy.

Our main contributions are threefold.

- We propose the Knowledge Distillation Based Fine-Tuning (KDFT), which fine-tunes channel-pruned DCNNs by any channel pruning methods, with no extra information and negligible computing overhead.
- We present a dynamic learning schedule to automatically set the hyper-parameter α according to the compression rate, which could alleviate the hyper-parameter sensitivity of knowledge distillation.
- We analyze the impact of structural skewness on KDFT, to guide practitioners to apply KDFT better under various practical scenarios.

Experiments are conducted on two benchmark datasets CIFAR-10 and CIFAR-100[27], using three representative large CNN models, including ResNet[3], VGGNet[28], and GoogLeNet[2]. The results demonstrate both the effectiveness and efficiency of our proposed method, compared with many state-of-the-art channel pruning methods.

2 Related Work

2.1 Non-Structural Model Compression

To remove redundant components in a neural network, most studies cared about fully-connected layers at first. Guo *et al.*[12] proposed a dynamic network surgery to remarkably reduce the network complexity by making on-the-fly connection pruning, which could recover timely when the pruning decision goes wrong. Srinivas *et al.*[29] pruned unimportant connections in neural networks by explicitly imposing sparse constraint over weights and stored the model in a sparse format after pruning. However, the speedup of these methods are based on the support of sophisticated sparse matrix operation libraries or hardware.

2.2 Structural Model Pruning

From Multilayer Perceptron (MLP)[30] to DCNN, the structure of deep neural network becomes increasingly modularized. This makes it more beneficial to prune whole convolutional channels, rather than pruning fully-connected layers.

Structured sparsity learning (SSL) presented by Wei *et al.*[13] made an effort to get a compact sparse model during training by adding a group Lasso constraint to the objective function and pruning channels with near-zero weights. Gao *et al.*[17] proposed a discrete model compression method, which prunes channels by measuring the overall discriminative pow-

er of a sub-network after closing certain discrete gates. These gates are inserted after feature maps, and the opening of gates is controlled by a learnable parameter θ. Other studies mainly focused on convolutional channel itself, looking for a good importance assessment mechanism to make pruning decisions. Lin *et al.*[16] found that the average rank of feature maps generated by a certain filter is nearly the same, regardless of how much data the CNN has seen. Thus the filters in a certain layer could be sorted and then pruned according to their average rank by scanning a number of input samples. Instead of convolutional filters, Liu *et al.*[18] focused on the batch normalization layers after convolutional layers. They indirectly assessed the importance of convolutional filters by their scaling factor produced by batch normalization layers. Chin *et al.*[31] proposed a global ranking method for filters in different layers, by using layer-wise affine transformations over filter norms to construct a global ranking of filters in a learnable fashion.

2.3 Knowledge Distillation

Hinton *et al.*[26] put forward the idea of teacher-student model for knowledge distillation. This method makes the student model learn from ground truth and the experience of teacher model at the same time, by introducing the softened output of the softmax layer of the teacher network as a soft target to the loss function of the student model. To make the distillation more suitable for deep models, Romero *et al.*[32] proposed Fitnets, which introduces an intermediate output of the teacher model as hint in addition to the softened logits. Yim *et al.*[33] proposed flow of the solution procedure (FSP) to train students for different tasks. FSP is a more complex knowledge representation which describes the transforming process of feature map outputs of different stages of deep structural models. Chen *et al.*[6] improved the accuracy of the student model by utilizing maximal teacher information through a review mechanism.

3 Knowledge Distillation Based Fine-Tuning

There are two stages in our method: 1) removing unimportant convolutional channels of a DCNN model with any channel pruning method, 2) performing knowledge distillation from the original model to the pruned model for fine-tuning, to recover the pruned model's accuracy effectively.

3.1 KDFT for Generic CNN Channel Pruning Methods

For a generic channel pruning method, no matter it prunes channels layer-wise or globally, the pruning process for a certain layer is similar. As shown in Fig.1, for the i-th convolutional layer, the three main parts of the network structure are input feature map $\boldsymbol{A}_i$, output feature map $\boldsymbol{A}_{i+1}$, and convolutional filters $\boldsymbol{K}_i$. Input $\boldsymbol{A}_i \in \mathbb{R}^{n_i,\ h_i,\ w_i}$ here is a 3D vector, with n_i channels of $h_i \times w_i$ 2D feature maps. $\boldsymbol{K}_i \in \mathbb{R}^{n_i,\ n_{i+1},\ h,\ w}$ represents n_{i+1} convolution filters containing n_i channels of $h \times w$ kernels. By applying these n_{i+1} convolutional filters on n_i channels of input feature maps in $\boldsymbol{A}_i$, we get n_{i+1} channels of feature maps in $\boldsymbol{A}_{i+1}$ after convolutional transformation.

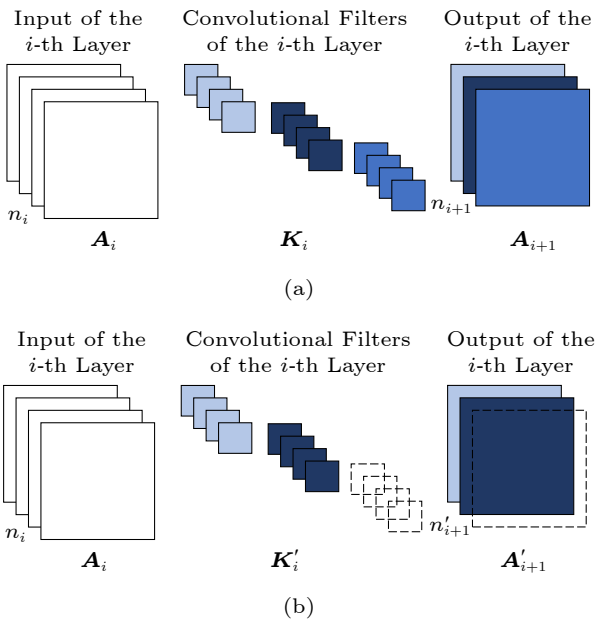

Fig.1. Illustration for channel pruning of the i-th convolutional layer. (a) Filters and feature maps before pruning. (b) Filters and feature maps after pruning.

With a pruning method, $F()$ (assuming it prunes layer by layer), we can prune a certain number of unimportant channels according to the compression rate for this layer, cp_i. We get the pruned kernel matrix $\boldsymbol{K}'_i \in \mathbb{R}^{n_i,\ n'_{i+1},\ h,\ w} = F(\boldsymbol{K}_i,\ cp_i)$. We can see from Fig.1 that if we prune one convolutional filter away, we also reduce one channel of the output feature maps $(n'_{i+1} = n_{i+1} - 1)$ of the i-th layer, and the channel dimension of the filters in the next layer is also reduced by one.

After pruning all layers in the original pre-trained model, we get a pruned model. Denoting them as M_{T} and M_{S}, respectively, we follow the settings in [26] to

illustrate the fine-tuning process of KDFT with M_{T} and M_{S}. For a network input sample $\boldsymbol{x}^i$, we can get two feature vectors extracted by the teacher and student networks as $\boldsymbol{z}_{\mathrm{T}}^i = N_{\mathrm{T}}(\boldsymbol{x}^i)$ and $\boldsymbol{z}_{\mathrm{S}}^i = N_{\mathrm{S}}(\boldsymbol{x}^i)$, respectively. These feature vectors, called logits, are actually the input of the softmax layer, while N_{T} and N_{S} are the teacher network and the student network without softmax, respectively. Through the softmax layer, the logits vectors $\boldsymbol{z}^i$ ($\boldsymbol{z}_{\mathrm{T}}^i$ or $\boldsymbol{z}_{\mathrm{S}}^i$) could be converted to the possibility for each class.

$$\boldsymbol{q}^i = \frac{\exp(\boldsymbol{z}^i)}{\sum_j \exp(\boldsymbol{z}_j^i)}.$$

Since there are significant differences between the teacher and student networks, logits computed by them might not be similar. A temperature parameter TP is introduced to the softmax function to soften the output of softmax, so that the student could learn from the teacher more conveniently.

$$\boldsymbol{q}_k^i = \frac{\exp(\boldsymbol{z}_k^i/TP)}{\sum_j \exp(\boldsymbol{z}_j^i/TP)}.$$

With the guidance of the teacher network, the loss function of the student network is written as

$$L = \frac{1}{n}\sum_{i=1}^{n}((1-\alpha)D(\boldsymbol{q}_{\mathrm{S}}^i, \boldsymbol{y}^i) + \alpha TP^2 D(\boldsymbol{q}_{\mathrm{S}}^i, \boldsymbol{q}_{\mathrm{T}}^i)), \quad (1)$$

where $\boldsymbol{y}^i$ is the ground truth label of the i-th input sample, $D(\cdot,\cdot)$ is the cross entropy loss for guaranteeing the performance of the student network. The left term in the function is the hard target, which measures the distance between student network outputs and the ground truth. The right term is the soft target and calculates the cross entropy between the student and teacher outputs. The weight parameter α is used to balance the soft and hard targets, make sure that M_{S} learns a good proportion from the ground truth and M_{T}.

During the training process of M_{S}, the network of M_{T} is frozen and batches of input samples from the training set are fed into both networks, M_{T} only has to inference from input and provide guidance with logits, and the loss of M_{S} is calculated according to (1). Then the error is back-propagated only on the student network. A detailed algorithm pseudocode for KDFT is shown in Algorithm 1. For every fine-tuning epoch, parameter α is automatically set by function $Dynamic\_alpha()$, which will be detailed in Subsection 3.2 and the pruned model is fine-tuned with knowledge distillation (KD).

Algorithm 1. KDFT

Input: channel pruning method F, compression rate cp, DCNN M, maximum fine-tuning epochs E_m

Output: fine-tuned channel-pruned DCNN M_{KDFT}

```
M' = F(M, cp);
M_KDFT = M';
for E = 1, 2, ..., E_m do
    α = Dynamic_alpha(E, cp);
    M_KDFT = KD(M, M_KDFT, α);
end for
return M_KDFT;
```

3.2 Fine-Tuning Pruned ResNets with HRank Analysis

To illustrate the effectiveness of KDFT and analyze the impact of parameter α, we employ our method on the channel pruning method HRank[16]. Lin *et al.*[16] applied their method to many deep networks such as VGG-16[28], GoogLeNet[2], and MobileNet[34]. We only fine-tune the pruned model of ResNet56[3] because its architecture is deeper and more complex, which makes it more representative.

The original ResNet56 model is trained with the dataset CIFAR-10[27], with an accuracy of 93.12. Firstly we apply HRank to the pre-trained model with three different compression rates, 22.4%, 42.9%, and 71.8%, respectively. Then we fine-tune the pruned models primitively for 150 epochs, and the recovered accuracies of the three models are 93.29%, 92.90%, and 91.52%, respectively, which is illustrated in Fig.2. We can see that the recovery results are not bad, the accuracy of the pruned model with low compression rate even surpasses the original model. One explana-

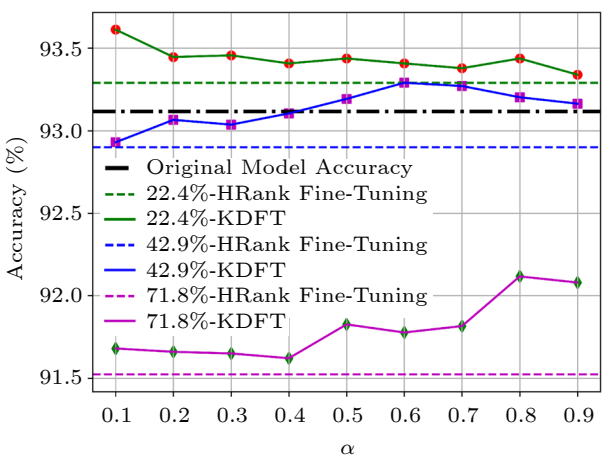

Fig.2. Accuracy of ResNet56 models after pruning by HRank and fine-tuning by KDFT, with different α values. Dash-dotted and dashed lines indicate the accuracy of original model and the model fine-tuned without KDFT, respectively. The legend indicates the compression rate and the fine-tuning method of a curve.

tion for this phenomenon is that the pruned model has less parameters than the original model to a proper extent, which does not harm its fitting ability but makes it easier to train under this circumstance.

To analyze the impact of parameter α, for every compression rate, we try different values from 0.1 to 0.9 with a stride of 0.1. The results are presented in Fig.2. In general, the performance of KDFT is always better than that of the primitive fine-tuning. It can be seen that the directions of the lines above the original model accuracy are opposite to the lines below it. When the recovered model accuracy is higher than that of the original model, it descends with the increase of α. And on the contrary, when the recovered model accuracy is lower than that of the original model, it ascends with the increase of α. This phenomenon is very intuitive: when the student is naive, learning from the teacher is easier for it than learning from the world directly. But when the student is better than its teacher, knowledge and experience from the teacher could only restrict the development of the student, and it is better to explore the world itself.

In order to make sure that the student network learns enough from the teacher network but is not restricted by it, we propose a dynamic learning schedule, with an adaptive parameter α, according to the compression rate of the pruned model. Let E be the current epoch, E_m be the total fine-tuning epochs and cp denotes the parameter compression rate of the pruned model. We get a dynamic learning schedule $Dynamic\_alpha()$ following a linear piecewise function as below.

$$\alpha = \begin{cases} 0.9, & \text{if } E < E_m \times cp, \\ 0.9 - \dfrac{0.8 \times (E - E_m \times cp)}{0.9 \times E_m - E_m \times cp}, & \\ & \text{if } E_m \times cp < E < 0.9 \times E_m, \\ 0.1, & \text{if } E > 0.9 \times E_m. \end{cases}$$

4 Experiments

4.1 Settings

Datasets and Models. To demonstrate the effectiveness and efficiency of KDFT on channel pruned methods for DCNN pruning, we conduct experiments on two benchmark datasets, CIFAR-10 and CIFAR-100[27], for image classification. The performance of KDFT on many popular DCNN models is studied, including ResNet56 and ResNet34 with residual blocks, VGG-16, and GoogLeNet with inception mudules. Resnet34 is trained and fine-tuned on CIFAR-100, while other models on CIFAR-10.

Metrics. To evaluate the pruning results quantitatively, we use two metrics, Params (number of parameters) and FLOPs (float Point operations) to represent model size and its needed computing resource. Besides, PR (pruning rate) is adopted to show how much a model is pruned. If not specified, the parameter reduction denotes the model compression rate. Top-1 accuracy is used as the metric for model accuracy.

Implementations. Our method KDFT is applied to three channel pruning methods, HRank[16], BN-slimming[18], and LeGR[31], implemented by PyTorch[35]. Before pruning, HRank uses 10 input samples to calculate the average rank of each convolutional channel, and BN-slimming needs a sparse training for 200 epochs to prepare for the pruning. In the fine-tuning stage, all these methods use the optimizer of the Stochastic Gradient Descent algorithm (SGD)[36]. All the hyper-parameters for fine-tuning, including initial learning rate, batch size, weight decay, learning decay step, and momentum are set according to [16, 18, 31], respectively. All experiments are conducted on two NVIDIA GTX 3060 GPUs.

4.2 Results on CIFAR-10

CIFAR-10 is a dataset containing 10 classes of images, and there are 50k training and 10k testing images, whose resolution is 32×32. Here we train and prune three DCNN models, ResNet56, VGG-16, and GoogLeNet with HRank, and the first two with BN-slimming and LeGR[31]. The reason we choose these three methods is that they are representative, since they represent different visions for pruning. HRank sorts the channels in every layer according to their average rank given a certain number of input samples, and it prunes away unimportant channels layer by layer, thus the compression rate for every layer in controllable. BN-slimming sorts all the convolutional channels in the network according to their scale factor and prunes channels globally, which may lead to extreme pruning in some layers. L2-norm[13] is a traditional metric to compare the importance of different channels with each convolutional layer, and LeGR proposes a learned affine transformation to improve the applicability of L2-norm, making it suitable to compare channels across different layers. To conclude, these methods have different assessment angles for filters. LeGR focuses directly on the convolutional fil-

ters and compares their L2-norms, while HRank and BN-slimming use indirect means to assess the importance of filters.

Comparison with HRank. Table 1 shows the comparison results on CIFAR-10. The models are pruned by HRank, fine-tuned with and without KDFT, respectively. The three original models are trained for 200 epochs and every model is fine-tuned for 300 epochs after pruning. The learning rate is initially set to 0.01, and divided by 10 at epochs 150 and 225. The network structures of the three DCNNs are different from each other, but as shown in Table 1, the performance of KDFT is always better than that of primitive fine-tuning, and the accuracy improvements for different compression rates nearly follow the same trend. Taking ResNet56 for example, for the compression rate of 22.4%, 42.9%, and 71.8%, the accuracy of the models fine-tuned by KDFT exceeds HRank by 0.07%, 0.46%, and 0.50%, respectively. KDFT could even improve the fine-tuning accuracy when the student's performance is better than that of the teacher model. And a same trend followed by all models is that when the compression rate is higher, the improvement of KDFT is more obvious (e.g., $0.07 < 0.46 < 0.50$). That is a great news since the bottleneck of improving compression rate for channel pruning methods is exactly the unbearable drop of accuracy. With KDFT, the compression limit of pruning methods could be pushed to a further extent.

Comparison with BN-Slimming. The results of pruning and fine-tuning comparison with BN-slimming is shown in Table 2. The original models are trained with channel sparse regularization for 200 epochs. The learning rate is initially set to 0.1, and divided by 10 after every 50 epochs. The training parameters for the fine-tuning stage are the same as training stage. We can see the fine-tuning accuracy improvements by KDFT is more significant on models pruned by BN-slimming. For ResNet56 with compression rate of 84.7%, pruned by BN-slimming, the improvement is 0.83%, while the one pruned with HRank by 71.8% compression rate is improved by 0.50%. As for VGG-16-BN models, the HRank pruned model with compression rate of 83.3% is improved by 0.08% through KDFT, while the improvement of the model 83.5% pruned by BN-slimming is 0.47%. We hypothesize this is due to the nature of the two methods. The models pruned by HRank are more structurally balanced since the compression rate is well controlled in every layer, and thus it is easier to train in the fine-tuning stage, while the model globally pruned by BN-slimming might be extremely pruned in some layers and its accuracy is harder to recover. This may be the reason that knowledge distillation has more effect on the models pruned by BN-slimming.

Comparison with LeGR. LeGR is a globally pruning method like BN-slimming, which may also end up with an unbalanced structure. We use ResNet56 and

Table 1. Fine-Tuning Comparisons Between KDFT and HRank[16] on CIFAR-10

Model	Params ($\times 10^6$)/PR (%)	FLOPs ($\times 10^6$)/PR (%)	HRank (%)	KDFT (%)
ResNet-56	0.85/0.0	126.55/0.0	93.12	–
	0.66/22.4	90.85/28.2	93.88	93.95
	0.49/42.9	65.94/47.9	93.39	93.85
	0.24/71.8	34.79/72.5	92.16	92.66
VGG-16-BN	14.99/0.0	314.29/0.0	93.94	–
	2.77/81.5	131.17/58.1	93.68	93.82
	2.51/83.3	104.78/66.7	93.63	93.71
	1.90/87.3	66.95/78.7	93.20	93.42
GoogLeNet	6.17/0.0	1 529.42/0.0	94.97	–
	2.86/53.6	649.19/57.6	94.90	95.10
	2.10/66.0	395.42/74.1	94.42	94.84

Table 2. Fine-Tuning Comparisons Between KDFT and BN-Slimming[18] on CIFAR-10

Model	Params ($\times 10^6$)/PR (%)	FLOPs ($\times 10^6$)/PR (%)	BN-Slimming (%)	KDFT (%)
ResNet-56	0.85/0.0	126.55/0.0	93.78	–
	0.69/18.1	89.89/29.0	92.66	93.99
	0.62/27.9	76.93/39.2	92.59	93.89
	0.13/84.7	16.56/86.9	87.27	88.10
VGG-16-BN	14.99/0.0	314.29/0.0	93.58	–
	4.64/76.9	197.80/37.0	93.76	93.97
	2.48/83.5	106.25/66.2	93.84	94.31

VGG-16-BN as the base models, which are trained for 200 epochs before pruning. In the fine-tuning stage, the initial learning rate is set to 0.01, and it is multiplied by 0.2 at the 18th, 36th, and 48th epoch, respectively, with totally 60 fine-tuning epochs. The accuracy improvement of LeGR is still larger than that of HRank, when the compression rates of both methods are near. For example, the accuracy improvement of ResNet56 pruned by HRank with compression rate 71.8% is 0.50%, while pruned by HRank with 75.6% compression rate, the improvement is 0.98%. This also supports our hypothesis that KDFT is more suitable for structurally extremely pruned models.

Comparison with Pure Knowledge Distillation. To illustrate the superiority of DKFT over the traditional knowledge distillation method, we conduct contrast experiments on models pruned by LeGR. For KDFT, we fine-tune the pruned model directly for 60 epochs, and for KD, the pretrained model before pruning is set to be the teacher, and the pruned model is set to be the student after we reinitialize the parameters of it, after that KD is trained for also 60 epochs. The results are shown in Table 3. We could see that with the same training budget, the performance of KDFT is always better than that of traditional KD, for both DCNN models and all compression rates.

4.3 Results on CIFAR-100

Besides experiments on the CIFAR-10 dataset, in this subsection we consider ResNet34 on the image classification task of CIFAR-100. Though shallower than ResNet56, ResNet34 has much more convolutional channels structurally, which makes it a better option for the classification task with more classes. Since there are 100 object classes in CIFAR-100, it is more challenging for DCNNs to get a good performance. The ResNet34 models are trained for 200 epochs and the accuracy of the pre-trained model on this dataset is only about 71%, which is much lower than that on the CIFAR-10 dataset. We prune the pre-trained model with HRank, BN-slimming, and LeGR at two compression rates respectively, afterwards the pruned models are fine-tuned with and without KDFT, respectively, and the results are shown in Table 4.

Every pruned model is fine-tuned for 200 epochs in Table 4. The accuracy improvement of KDFT is greater on the dataset CIFAR-100, with all pruning methods and all compression rates. With two different compression rates for the three methods respectively, the accuracies of models pruned by HRank are promoted by 0.96% and 0.29%, LeGR promotes their accuracies by 0.67% and 1.22%, while the accuracies of models pruned by BN-slimming are promoted by 2.89% and 3.36%, respectively. These improvements are of great significance for a 100-class classification task, while the overhead of KDFT is negligible. This fully demonstrates the effectiveness of KDFT. To confirm our hypothesis that KDFT works better on pruned models with unbalanced structures, we set a

Table 3. Fine-Tuning Comparisons Between KDFT and LeGR[31] on CIFAR-10

Model	Params ($\times 10^6$)/PR (%)	FLOPs ($\times 10^6$)/PR (%)	LeGR (%)	KDFT (%)	KD (%)
ResNet-56	0.85/0.0	126.55/0.0	92.87	–	–
	0.46/46.5	62.38/50.1	92.70	93.40	91.23
	0.33/61.0	37.45/70.0	91.34	92.30	89.97
	0.21/75.6	24.94/80.0	89.97	90.95	89.17
VGG-16-BN	14.99/0.0	314.29/0.0	93.70	–	–
	3.05/79.7	187.75/40.3	93.43	93.74	92.52
	2.39/84.1	156.42/50.2	93.44	93.79	92.45

Table 4. Fine-Tuning Pruned ResNet34 on CIFAR-100 with/Without KDFT

Method	Params ($\times 10^6$)/PR (%)	FLOPs ($\times 10^6$)/PR (%)	Accuracy (%)	KDFT (%)
HRank	21.15/0.0	1 155.10/0.0	71.15	–
	3.99/81.1	298.56/74.2	72.94	73.90
	2.76/87.0	208.97/81.3	72.71	73.00
BN-slimming	21.14/0.0	1 115.10/0.0	71.12	–
	4.78/77.4	357.7/68.0	67.13	70.02
	3.81/83.3	285.09/74.4	66.19	69.55
LeGR	21.14/0.0	1 115.10/0.0	77.40	–
	7.99/62.2	575.69/48.4	76.46	77.13
	4.18/80.2	288.16/74.2	74.00	75.22

constrain that at least 15% channels in each layer should be preserved in the pruning stage. Under this circumstance, the improvement by KDFT with BN-slimming is distinctly more remarkable than that with HRank and LeGR.

4.4 More Analysis

Efficiency Analysis. The accuracy recover effectiveness of KDFT is evidently better than primitive fine-tuning, which is already proved by the experiments above. In this subsection, we consider the efficiency instead, to demonstrate how fast KDFT could be, to get a better recovered accuracy. Taking fine-tuning ResNet34 on CIFAR-100 for example, we measure how many epochs it takes for KDFT to get the same accuracy as or better accuracy than primitive fine-tuning. The pruning ratio and methods here are the same as in Table 4. After pruning, we fine-tune the pruned models for 100 epochs. In the first 20 epochs, the learning rate is set to 0.1, and 0.01 afterwards. For KDFT, we use the same learning rate schedule, but when the fine-tuned accuracy of KDFT reach or surpass the final accuracy of primitive fine-tuning, we record the epoch and accuracy of it. The results are displayed in Table 5.

The result of pruning 87% parameters with HRank shows that after primitive fine-tuning for 100 epochs, the accuracy of the pruned model is recovered to 67.76%, while it only takes 29 epochs for KDFT to get 67.87% and 100 epochs to get the final accuracy of 68.57%. That is nearly 70% time saving since the overhead for knowledge distillation is negligible. While the final accuracy is promoted with KDFT by 0.81% in terms of effectiveness.

The efficiency promotion of BN-slimming is more significant than that of HRank. For the two compression rates, 77.4% and 83.3%, it only takes 21% tuning time for KDFT to get a similar accuracy to primitive fine-tuning. The improvement in accuracy is even more obvious. KDFT improves the final accuracy for the two compression rates by 4.86% and 3.90%, respectively, which are great improvements for a 100-class image classification task.

Structural Skewness Analysis. To analyze the impact of structural disbalance between different convolutional layers after channel pruning, we conduct experiments on ResNet56 with LeGR. To show the stable influence of structural skewness, we set different channel preserving proportions for different compression rates, and every pruned model comes from a same pretrained model. We do not preserve too many channels for compression rates 70% and 80% because that would compromise the pruning decision by a global pruning method too much.

We can see from Table 6 that for every compression rate, when the channel preserving rate increases (i.e., with a more balanced structure), the accuracy after pruning and naive fine-tuning gets better. But as the skewness increases, the accuracy improvement by KDFT is higher. This verifies that a structurally unbalanced model is harder to fine-tune, but it could be easier when there is a good teacher model.

5 Conclusions

In this paper, we proposed a fine-tuning method KDFT for channel-pruned DCNN models. Inspired by the idea of knowledge distillation, we found the DCNN models before and after channel pruning make a perfect pair of the teacher and student, and the accuracy of pruned models could be better recovered with the help of teacher knowledge with minimum computing and storage overhead. We proposed a dynamic tuning schedule with adaptive parameter α to get rid of the hyperparametric influence. Experiments were conducted on CIFAR-10 and CIFAR-100, with three representative pruning methods and several state-of-the-art DCNN models. The results have shown the effectiveness and efficiency (up to 79% time saving) of KDFT.

Different from traditional knowledge distillation models, there are many shared structures and parameters between the models before and after pruning. In the future, we aim to develop specific methods to effectively distill knowledge between structure sharing models.

Conflict of Interest The authors declare that they have no conflict of interest.

Table 5. Epoch Comparison for Fine-Tuning ResNet34 on CIFAR-100

Method	Params ($\times 10^6$)/ PR(%)	FLOPs ($\times 10^6$)/PR (%)	Accuracy (%)	KDFT (%)	Breaking Epoch/Accuracy (%)
HRank	3.99/81.1	298.56/74.2	69.38	69.47	38/69.47
	2.76/87.0	208.97/81.3	67.76	68.57	29/67.87
BN-slimming	4.78/77.4	357.7/68.0	62.52	67.38	21/64.63
	3.81/83.3	285.09/74.4	62.90	66.80	21/64.11

Table 6. Results of Skewness Analysis

Cp_rate	Preserve (%)	Accuracy (%)	Δ_{acc} (%)
50	0	92.73	0.45
	5	92.81	0.32
	10	92.85	0.20
	15	92.92	0.15
70	0	90.70	0.56
	5	90.82	0.39
	10	90.92	0.17
80	0	88.84	0.82
	5	89.25	0.70
	10	90.36	0.16

Note: The ResNet56 models are pruned by LeGR on CIFAR-10, with different compression rates and channel preserving rates. Preserve here indicates the minimum percentage of channels to be reserved in each layer, while Cp_rate is short for compression rate, and Δ_{acc} denotes the accuracy improvement by KDFT.

References

[1] Krizhevsky A, Sutskever I, Hinton G E. ImageNet classification with deep convolutional neural networks. *Communications of the ACM*, 2017, 60(6): 84–90. DOI: 10.1145/3065386.

[2] Szegedy C, Liu W, Jia Y, Sermanet P, Reed S, Anguelov D, Erhan D, Vanhoucke V, Rabinovich A. Going deeper with convolutions. In *Proc. the 2015 IEEE Conference on Computer Vision and Pattern Recognition*, Jun. 2015, pp.1–9. DOI: 10.1109/cvpr.2015.7298594.

[3] He K, Zhang X, Ren S, Sun J. Deep residual learning for image recognition. In *Proc. the 2016 IEEE Conference on Computer Vision and Pattern Recognition*, Jun. 2016, pp.770–778. DOI: 10.1109/cvpr.2016.90.

[4] Niyaz U, Bathula D R. Augmenting knowledge distillation with peer-to-peer mutual learning for model compression. In *Proc. the 19th International Symposium on Biomedical Imaging*, Mar. 2022, pp.1–4. DOI: 10.1109/ISBI52829.2022.9761511.

[5] Morikawa T, Kameyama K. Multi-stage model compression using teacher assistant and distillation with hint-based training. In *Proc. the 2022 IEEE International Conference on Pervasive Computing and Communications Workshops and Other Affiliated Events*, Mar. 2022, pp.484–490. DOI: 10.1109/PerComWorkshops53856.2022.9767229.

[6] Chen P, Liu S, Zhao H, Jia J. Distilling knowledge via knowledge review. In *Proc. the 2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, Jun. 2021, pp.5006–5015. DOI: 10.1109/cvpr46437.2021.00497.

[7] Redmon J, Divvala S, Girshick R, Farhadi A. You only look once: Unified, real-time object detection. In *Proc. the 2016 IEEE Conference on Computer Vision and Pattern Recognition*, Jun. 2016, pp.779–788. DOI: 10.1109/CVPR.2016.91.

[8] Ren S, He K, Girshick R, Sun J. Faster R-CNN: Towards real-time object detection with region proposal networks. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 2017, 39(6): 1137–1149. DOI: 10.1109/tpami.2016.2577031.

[9] Shelhamer E, Long J, Darrell T. Fully convolutional networks for semantic segmentation. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 2017, 39(4): 640–651. DOI: 10.1109/tpami.2016.2572683.

[10] Chen L C, Papandreou G, Kokkinos I, Murphy K, Yuille A L. DeepLab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected CRFs. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 2018, 40(4): 834–848. DOI: 10.1109/tpami.2017.2699184.

[11] Han S, Pool J, Tran J, Dally W J. Learning both weights and connections for efficient neural network. In *Proc. the 28th International Conference on Neural Information Processing Systems*, Dec. 2015, pp.1135–1143.

[12] Guo Y, Yao A, Chen Y. Dynamic network surgery for efficient DNNs. In *Proc. the 30th International Conference on Neural Information Processing Systems*, Dec. 2016, pp.1387–1395.

[13] Wen W, Wu C, Wang Y, Chen Y, Li H. Learning structured sparsity in deep neural networks. In *Proc. the 30th International Conference on Neural Information Processing Systems*, Dec. 2016, pp.2074–2082.

[14] Chen W, Wilson J T, Tyree S, Weinberger K Q, Chen Y. Compressing neural networks with the hashing trick. In *Proc. the 32nd International Conference on Machine Learning*, Jul. 2015, pp.2285–2294.

[15] Denton E, Zaremba W, Bruna J, LeCun Y, Fergus R. Exploiting linear structure within convolutional networks for efficient evaluation. In *Proc. the 27th International Conference on Neural Information Processing Systems*, Dec. 2014, pp.1269–1277.

[16] Lin M, Ji R, Wang Y, Zhang Y, Zhang B, Tian Y, Shao L. HRank: Filter pruning using high-rank feature map. In *Proc. the* 2020 *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, Jun. 2020, pp.1526–1535. DOI: 10.1109/cvpr42600.2020.00160.

[17] Gao S, Huang F, Pei J, Huang H. Discrete model compression with resource constraint for deep neural networks. In *Proc. the 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, Jun. 2020, pp.1896–1905. DOI: 10.1109/cvpr42600.2020.00197.

[18] Liu Z, Li J, Shen Z, Huang G, Yan S, Zhang C. Learning efficient convolutional networks through network slimming. In *Proc. the 2017 IEEE International Conference on Computer Vision*, Oct. 2017, pp.2755–2763. DOI: 10.1109/iccv.2017.298.

[19] He Y, Lin J, Liu Z, Wang H, Li L J, Han S. AMC: AutoML for model compression and acceleration on mobile devices. In *Proc the 15th European Conference on Computer Vision*, Sept. 2018, pp.815–832. DOI: 10.1007/978-3-030-01234-2_48.

[20] Le Cun Y, Denker J S, Solla S A. Optimal brain damage. In *Proc. the 2nd International Conference on Neural Information Processing Systems*, Jan. 1989, pp.598–605.

[21] Lawson C L, Hanson R J, Kincaid D R, Krogh F T. Basic linear algebra subprograms for Fortran usage. *ACM Trans. Mathematical Software (TOMS)*, 1979, 5(3): 308–323. DOI: 10.1145/355841.355847.

[22] Denil M, Shakibi B, Dinh L, Ranzato M, de Freitas N. Predicting parameters in deep learning. In *Proc. the 26th International Conference on Neural Information Processing Systems*, Dec. 2013, pp.2148–2156.

[23] Jiang W, Wang W, Liu S. Structured weight unification and encoding for neural network compression and acceleration. In *Proc. the 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, Jun. 2020, pp.3068–3076. DOI: 10.1109/cvprw50498.2020.00365.

[24] Hu H, Peng R, Tai Y W, Tang C K. Network trimming: A data-driven neuron pruning approach towards efficient deep architectures. arXiv: 1607.03250, 2016. https://arxiv.org/abs/1607.03250, Sept. 2024.

[25] Guo J, Ouyang W, Xu D. Multi-dimensional pruning: A unified framework for model compression. In *Proc. the 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, Jun. 2020, pp.1505–1514. DOI: 10.1109/cvpr42600.2020.00158.

[26] Hinton G, Vinyals O, Dean J. Distilling the knowledge in a neural network. arXiv: 1503.02531, 2015. https://arxiv.org/abs/1503.02531, Sept. 2024.

[27] Krizhevsky A. Learning multiple layers of features from tiny images. [Master Thesis], University of Toronto, 2009. https://www.cs.toronto.edu/~kriz/learning-features-2009-TR.pdf, Sept. 2024.

[28] Simonyan K, Zisserman A. Very deep convolutional networks for large-scale image recognition. arXiv: 1409.1556, 2015. https://arxiv.org/abs/1409.1556, Sept. 2024.

[29] Srinivas S, Subramanya A, Venkatesh Babu R. Training sparse neural networks. In *Proc. the 2017 IEEE Conference on Computer Vision and Pattern Recognition Workshops*, Jul. 2017. pp.455–462. DOI: 10.1109/cvprw.2017.61.

[30] Gardner M W, Dorling S R. Artificial neural networks (the multilayer perceptron)—A review of applications in the atmospheric sciences. *Atmospheric Environment*, 1998, 32(14/15): 2627–2636. DOI: 10.1016/S1352-2310(97)00447-0.

[31] Chin T W, Ding R, Zhang C, Marculescu D. Towards efficient model compression via learned global ranking. In *Proc. the 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, Jun. 2020. pp.1515–1525. DOI: 10.1109/cvpr42600.2020.00159.

[32] Romero A, Ballas N, Kahou S E, Chassang A, Gatta C, Bengio Y. FitnEts: Hints for thin deep Nets. In *Proc. the 3rd International Conference on Learning Representations*, May 2015.

[33] Yim J, Joo D, Bae J, Kim J. A gift from knowledge distillation: Fast optimization, network minimization and transfer learning. In *Proc. the 2017 IEEE Conference on Computer Vision and Pattern Recognition*, Jul. 2017, pp.7130–7138. DOI: 10.1109/cvpr.2017.754.

[34] Sandler M, Howard A, Zhu M, Zhmoginov A, Chen L C. MobileNetV2: Inverted residuals and linear bottlenecks. In *Proc. the 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, Jun. 2018. pp.4510–4520. DOI: 10.1109/cvpr.2018.00474.

[35] Paszke A, Gross S, Chintala S, Chanan G, Yang E, DeVito Z, Lin Z, Desmaison A, Antiga L, Lerer A. Automatic differentiation in Pytorch. In *Proc. the 31st Conference on Neural Information Processing Systems*, Dec. 2017.

[36] Robbins H, Monro S. A stochastic approximation method. *The Annals of Mathematical Statistics*, 1951, 22(3): 400–407. DOI: 10.1214/aoms/1177729586.

Chong Zhang is currently pursuing his Ph.D. degree in advanced manufacturing in the Faculty of Computing, Harbin Institute of Technology, Harbin. His research interests focus on deep model compression, development, and acceleration of neural networks on industrial applications, such as object detection.

Hong-Zhi Wang is a full professor in the Faculty of Computing, Harbin Institute of Technology, Harbin. He received his Ph.D. degree in computer science and technology from Harbin Institute of Technology, Harbin, in 2008. His research fields include big data management and analysis, database systems, knowledge engineering, and data quality.

Hong-Wei Liu is a full professor in the Faculty of Computing, Harbin Institute of Technology, Harbin. He received his Ph.D. degree in computer science and technology from Harbin Institute of Technology, Harbin, in 2004. His research interests mainly include computer system structure, cloud computing, and internet of things.

Yi-Lin Chen is currently an undergraduate student in Harbin Institute of Technology, Harbin. His research interests focus on the development of model compression and industrial applications of neural networks, such as object detection and instance segmentation.