ISSN: 2710-0170

HTCC: Combination Model Compression by Pruning and Tensor Decomposition

Dewen Wang[1, 2, a], Guanyue Wang[1, b, *]

[1]School of Control and Computer Engineering, North China Electric Power University, Baoding, Hebei, 071003, China

[2]Engineering Research Center of Intelligent Computing for Complex Energy Systems, Ministry of Education, Baoding, Hebei, 071003, China

[a]wdewen@gmail.com, [b, *]18295003058@163.com

Abstract

Convolutional neural network models have made great progress in various fields such as computer vision, Smart car driving, etc. However, the large number of weights in the model makes it difficult to be deployed in a low-memory environment. In order to solve this problem, researchers have considered single model compression methods such as pruning, quantization, and low-rank decomposition, Knowledge distillation and deployed them in practice. In this paper, we propose a hybrid model compression method. We prune the model by using the rank size of the output feature map as an important basis for discriminating filters, which can effectively consider the information of the entire network, and then perform Tucker decomposition on the convolutional layer to achieve further compression. We conduct experiments on the CIFAR10 dataset and show that Hrank Tucker Combination Compression (HTCC) achieves up to 85.8% over a single pruning and tensor decomposition, reducing Flops by 82.1%. At the same time, the accuracy of Resnet56 after compression is only reduced by less than 2%.

Keywords

Filter Pruning; Tensor Decomposition; Model Compression; Convolutional Neural Network.

1. Introduction

Convolutional neural networks have become an important tool in machine learning and many related fields [1]–[4]. With the continuous improvement of convolution neural network, convolution neural network has undergone many architectures and models, which achieve higher and higher accuracy. The earlier LeNet-5 [5]network was used for handwritten number recognition. Subsequently. AlexNet[6] reduced over fitting by using Relu activation function and dropout instead of regularization, and won the first place in ImageNet large-scale visual recognition challenge in 2012. In 2014, VGG [7]network proposed to replace a 7 * 7 convolution core with three 3 * 3 convolution cores to improve the resolution of the model by increasing the depth of the model. In 2015, GoogLeNet[8] adopts a new structure inception, and the error rate of its model is much lower than that of VGG on ImageNet dataset.

Model performance improvements are driven primarily by deeper and wider networks, but these usually slow down the execution speed, especially on resource-constrained devices. For example, the ResNet56[3] network with 56 convolution layers, involves more than 0.85M parameters for storage and over 125.49M floating point operations to process 32×32 color image. Therefore, on the premise of ensuring accuracy, it is necessary to reduce the complexity of the model. Model compression is an effective means to reduce the complexity of model, which includes model pruning, quantification, low rank decomposition, knowledge distillation. The

ISSN: 2710-0170

core of quantification is to convert 32-bit floating-point data types to fixed-point data types (16bit, 8bit, 4bit, and so on).At the extreme, the model weight can be binary(0,1)[9] or ternary (0,1,-1)[10].However, when quantifying to special bit widths (0, 1 and 0, -1, 1), many existing training methods and hardware platforms are no longer applicable, requiring a dedicated system architecture, which is not flexible. The core of knowledge distillation is to make the student's network better learn the advantages of the teacher's network, so as to achieve the effect of a large model with a small model. Since most knowledge distillation networks use the output of the softmax layer as knowledge, they are generally used for classification tasks with the softmax loss function, and the generalization of other tasks is not good. The essence of model pruning and tensor decomposition is to reduce the original parameters of the model. Model pruning removes redundant parameters by determining the importance of parameters. For example, Han Song et al. [10] set a threshold to zero the parameters below the threshold, convert the parameter matrix into a sparse matrix, and then retrain the network. Tensor decomposition reduces parameters by decomposing the convolution kernel tensor, replacing part with the whole, thus speeding up the model. For example, Denton et al. [11] approximates the weight matrix with a singular value score (SVD) and a filter clustering scheme, resulting in a two-fold acceleration with less than 1% loss of precision.

The above research mainly focus on a single operation of the model, which can not make full use of the advantages of various methods of model compression. In this paper, the mixed operation of model pruning and tensor decomposition is mainly focused on the redundancy of CNN model parameters. Compression model can combine the advantages of each model compression method, improve the compression rate of model compression and ensure the accuracy of the model.

In summary, the contributions of this study are as follows:

(a) We propose a two-step model compression method (Hrank Tucker combination compression, HTCC), that is, we first structure the pruning model, and then further compress the model by tensor decomposition.

(b) We compare the combination of filter pruning with different pruning rates and tensor decomposition with different thresholds. Finally, experiments show that the combined compression method with high pruning rate and low threshold has a good effect.

(c) We evaluate our method on CIFAR-10 data set, taking ResNet model as an example, we achieved a compression rate of up to 85.8%, reduced flops by 82.1%, and reduced resnet56 accuracy by less than 2%.

This article is organized as follows: we review the work related to model pruning and tensor decomposition in Sect. 2. In Sect.3, we describe a model compression method that combines filter pruning with tensor decomposition. In Sect.4, we conduct experiments under different configurations and related ablation experiments. We draw our conclusions and give the prospect of this study in Sect.5.

2. Related Work

2.1. Pruning

The core of model pruning is how to judge the importance of model parameters. The current mainstream model pruning methods are divided into two categories, one is unstructured pruning, the other is structured pruning. In earlier studies, Y Lecun [12] proposed Optimal Brain Damage (OBD), the basic idea of which is to use second derivatives to calculate the importance score of parameters, thereby removing the insignificant parameters. Recently, a Lottery Hypothesis theory has been proposed in [13]. Experiments show that there is an optimized sparse subnetwork structure in a more complex deep neural network, which can

ISSN: 2710-0170

be applied to model compression. Compared with the original network, the parameters and complexity of the sparse subnetwork are much lower, but the inference accuracy is basically the same. Malach et al.[14] have theoretically proved the lottery hypothesis, a fully over-parameterized neural network whose random initialization weight consists of a subnetwork which can be comparable to a complete network without additional training. All the above methods are based on gradually removing redundant neurons from the original large network to the small network. In [15], a greedy forward selection strategy is innovatively proposed, which builds a subnetwork by greedily adding the best neurons from the empty network. These methods belong to unstructured pruning, because the convolution core of unstructured pruning is sparse, that is, there are many matrices with zero elements in them. Sparse matrices have limitations in that they cannot leverage existing mature BLAS libraries to achieve additional performance benefits.

Due to the limitations of unstructured pruning, the focus of current research on model pruning is mainly on structured pruning. Filter pruning [16]–[19]is one of structured pruning. Ding et al. [17] uses centripetal SGD method to fold multiple filters into a single point in parameter space and delete the same after training. This achieves the purpose of model compression. He et al. [20]presents a soft filter pruning (SFP) method to accelerate the inference process of deep convolution neural networks. Specifically, the proposed SPF allows the trimmed filter to be updated when training the model after trimming, providing space for model optimization. In [21], a convolution kernel pruning (Filter Pruning via Geometric Median, FPGM) based on geometric median is proposed. It is worth noting that FPGM compresses CNN models by pruning redundant convolution cores instead of relatively small convolution cores. In recent years, with the rise of reinforcement learning, many researchers have derived a new model pruning method by combining it with model pruning. In [22], the author has designed an AMC (AutoML for Model Compression) automatic pruning framework by combining it with the reinforcement learning strategy. An automatic structured pruning framework AutoCompress based on Alternating Direction Method of Multipliers (ADMM) is proposed in [19].

2.2. Tensor Decomposition

The basic principle of tensor decomposition is to reorganize the weight matrix or tensor and use the low rank matrix or tensor to represent the original matrix by the decomposition method, thereby reducing the number of parameters. Denton et al. [23] approximate the weight matrix by singular value score (SVD) and filter clustering scheme, and achieves a 2-fold acceleration with less than 1% loss of accuracy. Zhang el at.[24] considers the non-linear response of the CNN and presents an asymmetric reconstruction method to reduce the cumulative error in the multilayer approximation. In addition, a low rank regular decomposition technique has been developed in [25] and satisfactory results have been obtained on large networks. To make better use of the low-level structure of the weight tensor, researchers have also proposed tensor decomposition algorithms, such as CP decomposition [26], Tucker decomposition [27], Tensor Train decomposition[28], Block Term decomposition [29], to reduce network redundancy using a low-tensor hierarchy.

2.3. Combined Model Compression

Model compression techniques can be combined to achieve better compression results. For example, the combination of pruning and quantification, Ullrich et al. [30] based on the regularization item of Software weight sharing, implemented parameter quantization and parameter pruning during the model retraining process. Tung et al. [31] proposed the integrated compression and acceleration framework Compression learning by in parallel pruning-quantization (CLIP-Q). In combination of quantification and knowledge distillation, Polino et al. [32] proposed a quantitative training method with knowledge distillation loss.

ISSN: 2710-0170

There are floating point model and quantification model. The quantification model calculates the forward loss and calculates the gradient to update the floating point model. Before each forward calculation, the quantification model is updated with the updated floating point model. A combination of pruning and knowledge distillation, Xie et al.[33]proposed a method for extracting knowledge into a pruning model to reduce parameters.

3. Methodology

Our proposed method aims to explore a new combined model compression method, which further improves the compression rate by combining the advantages of pruning and tensor decomposition. In this section, we describe the details of the filter pruning method and the Tucker decomposition method used for model compression.

3.1. Filter Pruning

Filter pruning is a kind of structured pruning, and the method to judge the importance of filters is the core part of the filter pruning algorithm. Li et al. and He et al. judged the importance of the filter by the L1 norm criterion [34] and the L2 norm criterion [20]. Luo et al. [35] pruned the filter by the next level of statistical information. Inspired by Lin et al. [16], we use the rank of the output layer to judge the importance of the filter. For two successive convolution operations, deleting the first convolution filter results in the corresponding kernel being deleted in the next convolution, as shown in Fig 1. The different methods described above are essentially the same, so we can convert the filter pruning problem into an optimization problem based on specific rules, removing unimportant filters through specific rules. Assume a pre-trained CNN model has a set of K convolution layers, and C^i is the $i-th$ convolution layer. The parameters are represented as a set of 3D filters $W_{C^i}=\{w_1,w_2,...,w_{n_i}\}\in\mathbb{R}^{n_i\times n_{i-1}\times k_i\times k_i}$,where the $j-th$ filter is $w_j^i\in\mathbb{R}^{n_{i-1}\times k_i\times k_i}$, n^i represents the number of filters in the filter and k^i is the kernel size. The outputs of filters are represented as $O^i=\left\{o_1^i,o_2^i,o_3^i,...,o_{n^i}^i\right\}\in\mathbb{R}^{n_i\times g\times h_i\times w_i}$,where the $j-th$ feature map is denoted as $o_j^i\in\mathbb{R}^{g\times h_i\times w_i}$, g is the size of input images. h_i and w_i are the height and width of the feature map. In filters pruning, W_{C^i} can be split into two groups, one is reserved filter $J_{C^i}=\left\{w_{j_1^i}^i,w_{j_2^i}^i,w_{j_2^i}^i,...,w_{j_{n_{i1}}^i}^i\right\}$, the other is removed filter $U_{C^i}=\left\{w_{u_1^i}^i,w_{u_2^i}^i,w_{u_2^i}^i,...,w_{u_{n_{i1}}^i}^i\right\}$,where J_j^i and U_j^i are represented as the $j-th$ filters to be reserved and the $u-th$ filters to be removed. We use n_{i1} and n_{i2} as the number of important and non-important filters, respectively.

The purpose of filter pruning is to eliminate the unimportant filters in the neural network, which can be turned into an optimization problem:

$$\begin{aligned}&\min_{\delta_{ij}}\sum_{i=1}^{K}\sum_{j=1}^{n_i}\delta_{ij}L(w_j^i)\\&s.t.\sum_{j=1}^{n_i}\delta_{ij}=n_{i2}\end{aligned}\tag{1}$$

where δ_{ij} is an indicator which is 0 if w_j^i is grouped to J_{C^i} or 1 if w_j^i is grouped to U_{C^i} . $L(w_j^i)$ is a measure of the importance of filter input to CNN. Thus, an unimportant filter is deleted according to Eq. (1). Filter pruning is shown in Fig 1.

ISSN: 2710-0170

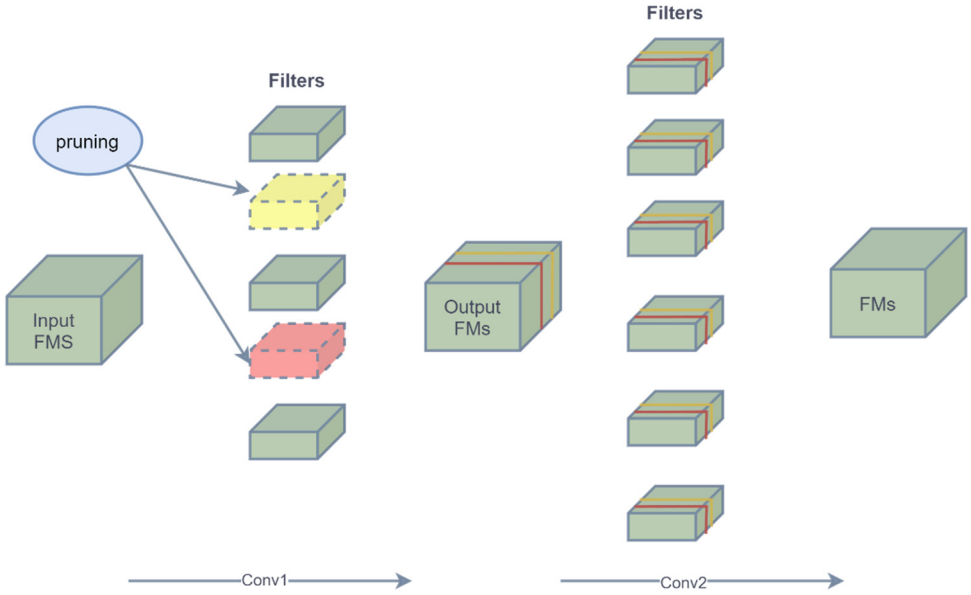

Fig 1. Filter pruning

3.2. Tucker Decomposition

Tensor decomposition is a concept based on linear algebra, which is called singular value decomposition (SVD). Tucker decomposition is actually a generalization of SVD [27]. In the study of model compression and acceleration, tensor decomposition has been widely used as the decomposition of convolution layer[26]–[29] and full connection layer[23] in CNN.

For CNN, the convolution layer can be regarded as a 4D tensor K with the size of $C_{out} \times C_{in} \times h \times w$, Where C_{in} and C_{out} represents the input of the convolution layer and the output of the convolution layer, and h, w denotes the spatial size of the tensor. The size of the input feature graph is expressed as $H \times W \times C_{in}$, and the size of the output feature graph is expressed as $H' \times W' \times C_{out}$. The rank - (R1, R2, R3, R4) Tucker of the 4D tensor (along all its modes) is decomposed into a core tensor G and a set of factor matrices U^1, U^2, U^3, U^4. As shown in Eq2:

$$K_{i,j,s,t} = \sum_{r_1=1}^{R_1} \sum_{r_2=1}^{R_2} \sum_{r_3=1}^{R_3} \sum_{r_4=1}^{R_4} G_{r_1,r_2,r_3,r_4} \times U^1_{i,r_1} \times U^2_{j,r_2} \times U^3_{s,r_3} \times U^4_{t,r_4} \tag{2}$$

where the size of G is $R_1 \times R_2 \times R_3 \times R_4$ and the size of factor matrix U^1, U^2, U^3, U^4 is $C_{out} \times R_1, C_{in} \times R_2, h \times R_3, w \times R_4$.

Decomposition along the spatial dimension can lead to spatially separable convolution, but in Tucker decomposition, we do not decompose model-1 and model-2 related to the spatial dimension [27], because most of their sizes are $1 \times 1, 3 \times 3, 5 \times 5$, separable convolution cannot save a lot of calculation. Therefore, the above Eq. (2) is converted to Eq. (3):

$$K_{i,j,s,t} = \sum_{r_3=1}^{R_3} \sum_{r_4=1}^{R_4} G'_{i,j,r_3,r_4} \times U^{3'}_{s,r_3} \times U^{4'}_{t,r_4} \tag{3}$$

where the size of G' is $R_3 \times R_4 \times h \times w$.

According to the above description, the changes of parameters and Floating-point operations (Flops) before and after convolution can be compared, as shown in Table 1.

ISSN: 2710-0170

Table 1. Comparison of parameters and Flops before and after Tucker decomposition

Type	Convolution	Tucker decomposition
parameters	$C_{out} \times C_{in} \times h \times w$	$R_3 \times R_4 \times h \times w + R_3 \times C_{in} + R_4 \times C_{out}$
Flops	$H' \times W' \times C_{in} \times C_{out} \times h \times w$	$H \times W \times R_3 \times C_{\text{in}} + H' \times W' \times R_3 \times R_4 \times h \times w$ $+H' \times W' \times C_{\text{out}} \times R_4$

Rank (R3, R4) is a very important Hyperparameter that controls the balance between performance (memory, speed, energy) improvement and accuracy loss[27]. In this paper, we use Variational Bayesian Matrix Decomposition (VBMF) to select rank. VBMF has been sufficiently validated in [27], [36] to provide better results than previous manual debugging and the use of alternating least squares.

3.3. Our Approach

We use a layer-by-layer pruning method to prune the filter, and determine the importance of the filter by the rank of the output layer. Lin et al. [16] show that a small number of images can effectively estimate the average rank of each output signature in different architectures, and then prune the filter based on the rank of the output signature. The proportion of filters removed requires a threshold parameter that provides a trade-off between space and accuracy by controlling the number of filters to trim. Since we use layer-by-layer pruning, we need to set the pruning rate for each layer manually. We experimented on a Cifar-10 dataset with an initial learning rate of 0.01, a time-varying 0.001 for the fifth and a 0.0001 for the tenth rounds. For each layer, we do 30 rounds of retraining after trimming. For each trimmed convolution layer, we use Tucker decomposition to further reduce the model parameters and computations. The compression mechanism of HTCC is shown in Fig 2.

For pruning, the input feature map size is $H \times W \times C_{in}$, the convolution layer can be represented as $C_{out} \times C_{in} \times h \times w$, the output feature map size is $H' \times W' \times C_{out}$, and the output feature size of the next layer is $H'' \times W'' \times C'_{out}$. When an unimportant set of filters a is removed U_{C^i} and an important set of filters J_{C^i} is retained, C_{out} converts C^J_{out} and the next convolution layer is represented as $C'_{out} \times C^J_{out} \times h' \times w'$.

Thus, the number of parameters is reduced from:

N: $C_{out} \times C_{in} \times h \times w + C'_{out} \times C_{out} \times h' \times w'$

to:

NP: $C^J_{out} \times C_{in} \times h \times w + C'_{out} \times C^J_{out} \times h' \times w'$

Regardless of the next layer of pruning, Tucker decomposition reduces the number of parameters to:

$$\text{NPT: } R_3 \times R_4 \times h \times w + R_3 \times C_{in} + R_4 \times C^J_{out} + R'_3 \times R'_4 \times h' \times w' + R'_3 \times C^J_{out} + R_4 \times C'_{out}$$

In fact, more parameters are reduced, because the filter pruning operation has been implemented for the next convolution layer before Tucker decomposition. In order to simplify the reasoning process, we do not make it clear. As the number of parameters decreases, the amount of Floating-point operations also decreases.

For pruning, Flops are reduced from:

$$\text{F: } \otimes\left(H' \times W' \times C_{out} \times C_{in} \times h \times w + H'' \times W'' \times C'_{out} \times C_{out} \times h' \times w'\right)$$

ISSN: 2710-0170

To:

$$\text{FP}: \otimes\left(H^{'}\times W^{'}\times C_{out}^{J}\times C_{in}\times h\times w+H^{''}\times W^{''}\times C_{out}^{'}\times C_{out}^{J}\times h^{'}\times w^{'}\right)$$

Regardless of the next layer of pruning, Tucker decomposition reduces Flops to:

$$\text{FPT}: \otimes\begin{pmatrix} H^{'}\times W^{'}\times R_3\times R_4\times h\times w+H\times W\times R_3\times C_{in}+H^{'}\times W^{'}\times R_4\times C_{out}^{J}+ \\ H^{''}\times W^{''}\times R_3^{'}\times R_4^{'}\times h^{'}\times w^{'}+H^{'}\times W^{'}\times R_3^{'}\times C_{out}^{J}+H^{''}\times W^{''}\times R_4\times C_{out}^{'} \end{pmatrix}$$

The compression ratio of decomposition is represented as:

$$M=\frac{N}{N_{PT}} \tag{4}$$

$$E=\frac{F}{F_{PT}} \tag{5}$$

Actually, M and E is smaller. Because we do not consider the next filter pruning. The compression process of HTCC is shown in Fig 3.

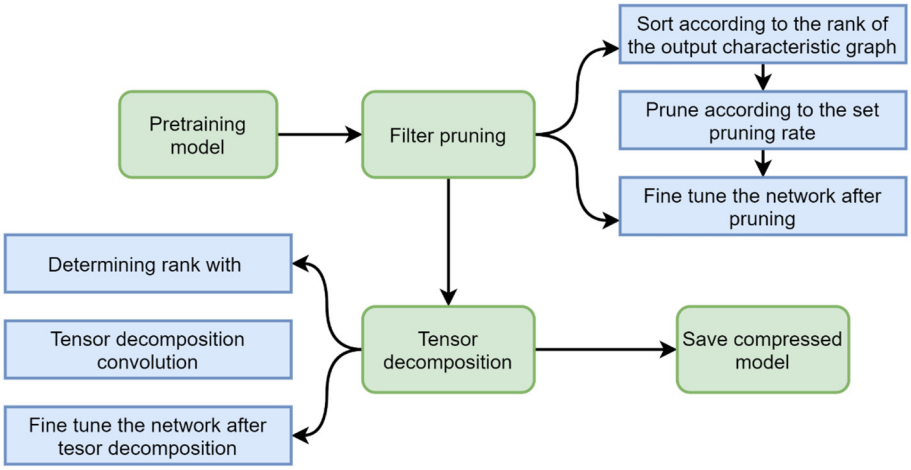

Fig 2. Compression mechanism of HTCC

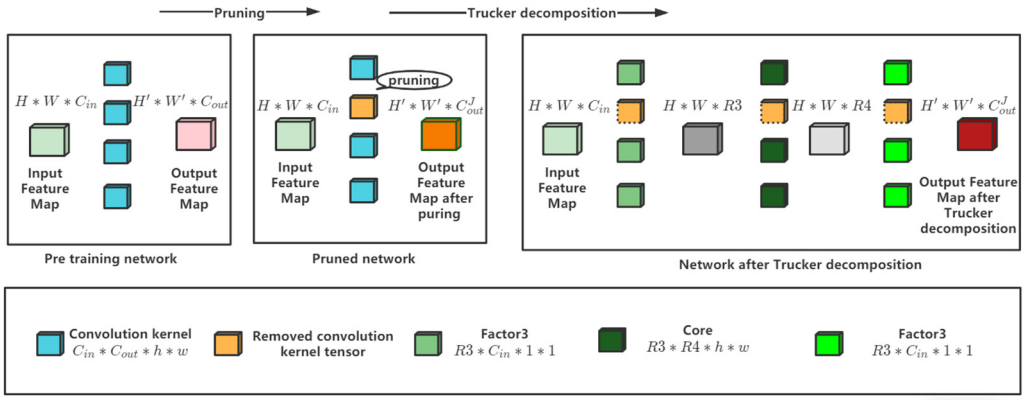

Fig 3. The process of the HTCC method, first pruning the pre-training network, then decomposing the Trucker, requires the final compressed network

ISSN: 2710-0170

4. Experiments

We analyze the performance on CIFAR-10, comparing against several popular CNNs, including ResNet56 and VGG-16.

4.1. Results on ResNet56

We validated our approach by training the ResNet56 network on the CIFAR-10 dataset. First, we controlled the pruning rate to 70% by adjusting the pruning threshold of each layer, resulting in a 1.55% reduction in model accuracy after pruning. By Tucker decomposition of the pre-training model alone, the parameter quantity of the model is reduced by 35.2%, and the calculation amount is reduced by 47.73%, but the accuracy of the model is only reduced by 0.22%. This may be because the higher rank has more useful information when choosing rank in VBMF. Tucker decomposition of the pruned model further compresses the model parameters to 85.8%, reduces the computational effort to 82.1%, and reduces the accuracy by only 1.89%. As shown in Table2.

Table 2. Compressing results of ResNet-56 on CIFAR-10
A: Pruning B: Tucker decomposition

Model	Params	Flops	Accuracy
Resnet5(baseline)	0.85M(0.0%)	125.49M(0.0%)	91.87%
Resnet56(A)	0.24M(70.0%)	34.78M(74.1%)	90.32% (-1.55%)
Resnet56(B)	0.55M(35.2%)	65.59(47.73%)	91.65% (-0.22%)
Resnet56(A+B)	0.12M(85.8%)	22.48M(82.1%)	89.98% (-1.89%)

To verify the advantages of HTCC, we compare it with the single pruning method and the single tensor decomposition method. The experimental results are shown in Table 3.

Table 3. Compressing results of ResNet-56 on CIFAR-10

Model	Params	Flops	Accuracy
Resnet56	0.85M(0.0%)	125.49M(0.0%)	91.87%
Hrank[16]	0.27M(68.1%)	40.31M(67.8%)	90.43%
GBN[37]	0.54M(53..5%)	75.41M(60.1%)	91.54%
Tucker[27]	0.55M(35.2%)	65.59(47.73%)	91.65%
HTCC(Ours)	0.12M(85.8%)	22.48M(82.1%)	89.98%

4.2. Results on VGG-16

We validated our approach by training the VGG-16 on the CIFAR-10 dataset. First, we controlled the pruning rate to 70% by adjusting the pruning threshold of each layer, resulting in a 1.08% reduction in model accuracy after pruning. By Tucker decomposition of the pre-training model alone, the parameter quantity of the model is reduced by 43.9%, and Flops is reduced by 32.27%, but the accuracy of the model is only reduced by 0.41%. Tucker decomposition of the pruned model further compresses the model parameters to 91.12%, reduces the computational effort to 72.05%, and reduces the accuracy by only 1.88%.To verify the advantages of HTCC, we compare VGG-16 with the single pruning method and the single tensor decomposition method. The experimental results are shown in Table 4 and Table 5.

ISSN: 2710-0170

Table 4. Compressing results of VGG-16 on CIFAR-10
A: Pruning B: Tucker decomposition

Model	Params	Flops	Accuracy
VGG-1(baseline)	14.98M(0.0%)	313.73M(0.0%)	92.76%
VGG-16(A)	4.49M(70.0%)	131.17M(58.1%)	91.68% (-1.08%)
VGG-16(B)	8.56M(43.9%)	212.47(32.27%)	92.35% (-0.41%)
VGG-16(A+B)	1.33M(91.12%)	87.69M(72.05%)	90.88% (-1.88%)

Our method was compared with Hrank [16]and GBN[37] pruning . Compared with Hrank, our parameters were reduced by 17.7%, Flops by 14.3%, and accuracy by only 0.45%. Compared with Tucker decomposition alone, the parameters were reduced by 50.6%, the Flops by 34.37, and the accuracy by 1.67%. The experimental results show that pruning and tensor decomposition alone do not fully reduce the number of parameters and operations. Our method combines the advantages of pruning and tensor decomposition to obtain a higher compression rate while ensuring accuracy. In addition, we validated our method on VGG-16, and the results show that in Table 4 and Table 5.

Table 5. Compressing results of VGG-16 on CIFAR-10

Model	Params	Flops	Accuracy
VGG-16	14.98M(0.0%)	313.73M(0.0%)	92.76%
Hrank[16]	2.88M(88.8%)	142.36M(54.2%)	91.73%(-1.03%)
GBN[37]	6.68M(55.4%)	180.39M(42.5%)	91.89%(-0.87%)
Tucker[27]	8.56M(43.9%)	212.47(32.27%)	92.35%(-0.41%)
HTCC(Ours)	1.33M(91.12%)	87.69M(72.05%)	90.88%(-1.88%)

4.3. Ablation Study

During the experiment, we found that different combinations of trimming and Tucker decomposition thresholds had biased results. As shown in Figure 4, Tucker decomposition with low thresholds and high pruning rate had better results.

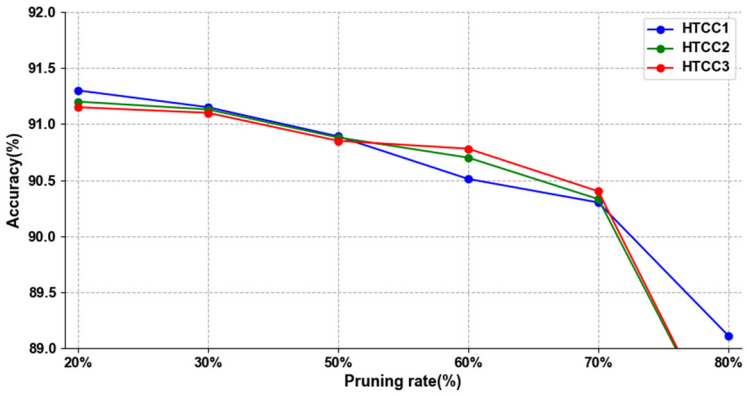

Fig 4. Various combinations of pruning and Tucker decomposition

ISSN: 2710-0170

HTCC1 indicates high threshold, HTCC2 indicates medium threshold and HTCC3 indicates low threshold. The overall graph shows that the accuracy of the model decreases with the increase of pruning rate, but HTCC3 performs better when the pruning rate is between 60% - 70%.

5. Conclusion

In this work, our experimental results show that compared with pruning and tensor decomposition alone, the combination of pruning and Trucker decomposition can obtain higher compression rate, less computation, and maintain good accuracy. At the same time, we experimentally verify that the Tucker decomposition combination with high pruning rate and low threshold has better comprehensive performance in the compression process. The main research focus in the future is to introduce the quantitative method into HTCC to obtain better compression performance.

Acknowledgments

This work is supported by the National Natural Science Foundation of China (No. 51677072) and the Fundamental Research Funds for the Central Universities(No. 2020MS120)and special thanks are given to the anonymous reviewers and editors for their constructive comments.

References

[1] N. Smolyanskiy, A. Kamenev, and S. Birchfield, “On the Importance of Stereo for Accurate Depth Estimation: An Efficient Semi-Supervised Deep Neural Network Approach.” pp. 1007–1015, 2018, Accessed: Aug. 01, 2021. [Online]. Available: https://youtu.be/0FPQdVOYoAU.

[2] GhoshSwarnendu, DasNibaran, DasIshita, and MaulikUjjwal, “Understanding Deep Learning Techniques for Image Segmentation,” ACM Comput. Surv., vol. 52, no. 4, Aug. 2019, doi: 10. 1145/ 3329784.

[3] K. He, X. Zhang, S. Ren, and J. Sun, “Deep Residual Learning for Image Recognition.” pp. 770–778, 2016, Accessed: Aug. 01, 2021. [Online]. Available: http://image-net.org/challenges/LSVRC/2015/.

[4] W.-L. Hsiao and K. Grauman, “Learning the Latent ‘Look’: Unsupervised Discovery of a Style-Coherent Embedding from Fashion Images,” Jul. 2017, Accessed: Aug. 01, 2021. [Online]. Available: http://arxiv.org/abs/1707.03376.

[5] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, “Gradient-based learning applied to document recognition,” Proc. IEEE, vol. 86, no. 11, pp. 2278–2323, 1998, doi: 10.1109/5.726791.

[6] KrizhevskyAlex, SutskeverIlya, and H. E., “ImageNet classification with deep convolutional neural networks,” Commun. ACM, vol. 60, no. 6, pp. 84–90, May 2017, doi: 10.1145/3065386.

[7] K. Simonyan and A. Zisserman, “Very Deep Convolutional Networks for Large-Scale Image Recognition,” 3rd Int. Conf. Learn. Represent. ICLR 2015 - Conf. Track Proc., Sep. 2014, Accessed: Aug. 01, 2021. [Online]. Available: https://arxiv.org/abs/1409.1556v6.

[8] C. Szegedy et al., “Going Deeper With Convolutions.” pp. 1–9, 2015.

[9] M. Rastegari, V. Ordonez, J. Redmon, and A. Farhadi, “XNOR-Net: ImageNet Classification Using Binary Convolutional Neural Networks,” Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics), vol. 9908 LNCS, pp. 525–542, 2016, doi: 10.1007/978-3-319-46493-0_32.

[10] F. Li, B. Zhang, and B. Liu, “Ternary Weight Networks,” May 2016, Accessed: Aug. 01, 2021. [Online]. Available: https://arxiv.org/abs/1605.04711v2.

[11] E. Denton, W. Zaremba, J. Bruna, Y. LeCun, and R. Fergus, “Exploiting Linear Structure Within Convolutional Networks for Efficient Evaluation,” Adv. Neural Inf. Process. Syst., vol. 2, no. January, pp. 1269–1277, Apr. 2014, Accessed: Aug. 01, 2021. [Online]. Available: https:// arxiv. org/ abs/ 1404. 0736v2.

ISSN: 2710-0170

[12] Y. LeCun, J. Denker, S. S.-A. in neural information, and undefined 1990, "Optimal brain damage," papers.nips.cc, Accessed: Aug. 02, 2021. [Online]. Available: http://papers.nips.cc/paper/250-optimalbrain-damage.pdf.

[13] J. Frankle and M. Carbin, "The Lottery Ticket Hypothesis: Finding Sparse, Trainable Neural Networks," 7th Int. Conf. Learn. Represent. ICLR 2019, Mar. 2018, Accessed: Aug. 02, 2021. [Online]. Available: https://arxiv.org/abs/1803.03635v5.

[14] E. Malach, G. Yehudai, S. Shalev-Schwartz, and O. Shamir, "Proving the Lottery Ticket Hypothesis: Pruning is All You Need." PMLR, pp. 6682–6691, Nov. 21, 2020, Accessed: Aug. 02, 2021. [Online]. Available: http://proceedings.mlr.press/v119/malach20a.html.

[15] M. Ye, C. Gong, L. Nie, D. Zhou, A. Klivans, and Q. Liu, "Good Subnetworks Provably Exist: Pruning via Greedy Forward Selection." PMLR, pp. 10820–10830, Nov. 21, 2020, Accessed: Aug. 02, 2021. [Online]. Available: http://proceedings.mlr.press/v119/ye20b.html.

[16] M. Lin et al., "HRank: Filter Pruning Using High-Rank Feature Map." pp. 1529–1538, 2020, Accessed: Aug. 02, 2021. [Online]. Available: https://github.com/lmbxmu/HRank.

[17] X. Ding, G. Ding, Y. Guo, and J. Han, "Centripetal SGD for Pruning Very Deep Convolutional Networks With Complicated Structure." pp. 4943–4953, 2019, Accessed: Aug. 02, 2021. [Online]. Available: https://github.com/.

[18] T.-W. Chin, R. Ding, C. Zhang, and D. Marculescu, "Towards Efficient Model Compression via Learned Global Ranking." pp. 1518–1528, 2020, Accessed: Aug. 02, 2021. [Online]. Available: https:// github. com/ cmu-enyac/LeGR.

[19] N. Liu, X. Ma, Z. Xu, Y. Wang, J. Tang, and J. Ye, "AutoCompress: An Automatic DNN Structured Pruning Framework for Ultra-High Compression Rates," Proc. AAAI Conf. Artif. Intell., vol. 34, no. 04, pp. 4876–4883, Apr. 2020, doi: 10.1609/AAAI.V34I04.5924.

[20] Y. He, G. Kang, X. Dong, Y. Fu, and Y. Yang, "Soft Filter Pruning for Accelerating Deep Convolutional Neural Networks," IJCAI Int. Jt. Conf. Artif. Intell., vol. 2018-July, pp. 2234–2240, Aug. 2018, Accessed: Aug. 02, 2021. [Online]. Available: https://arxiv.org/abs/1808.06866v1.

[21] Y. He, P. Liu, Z. Wang, Z. Hu, and Y. Yang, "Filter Pruning via Geometric Median for Deep Convolutional Neural Networks Acceleration." pp. 4340–4349, 2019, Accessed: Aug. 02, 2021. [Online]. Available: https://github.com/he-y/filter-pruning-geometric-median.

[22] Y. He, J. Lin, Z. Liu, H. Wang, L.-J. Li, and S. Han, "AMC: AutoML for Model Compression and Acceleration on Mobile Devices." pp. 784–800, 2018.

[23] E. Denton, W. Zaremba, J. Bruna, Y. LeCun, and R. Fergus, "Exploiting Linear Structure Within Convolutional Networks for Efficient Evaluation," Adv. Neural Inf. Process. Syst., vol. 2, no. January, pp. 1269–1277, Apr. 2014, Accessed: Aug. 02, 2021. [Online]. Available: https:// arxiv.org/ abs/ 1404. 0736v2.

[24] X. Zhang, J. Zou, X. Ming, K. He, and J. Sun, "Efficient and accurate approximations of nonlinear convolutional networks," Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit., vol. 07-12-June-2015, pp. 1984–1992, Oct. 2015, doi: 10.1109/CVPR.2015.7298809.

[25] C. Tai, T. Xiao, Y. Zhang, X. Wang, and W. E, "Convolutional neural networks with low-rank regularization," 4th Int. Conf. Learn. Represent. ICLR 2016 - Conf. Track Proc., Nov. 2015, Accessed: Aug. 02, 2021. [Online]. Available: https://arxiv.org/abs/1511.06067v3.

[26] V. Lebedev, Y. Ganin, M. Rakhuba, I. Oseledets, and V. Lempitsky, "Speeding-up Convolutional Neural Networks Using Fine-tuned CP-Decomposition," 3rd Int. Conf. Learn. Represent. ICLR 2015 - Conf. Track Proc., Dec. 2014, Accessed: Aug. 02, 2021. [Online]. Available: https:// arxiv.org /abs/1412. 6553v3.

[27] Y.-D. Kim, E. Park, S. Yoo, T. Choi, L. Yang, and D. Shin, "Compression of Deep Convolutional Neural Networks for Fast and Low Power Mobile Applications," 4th Int. Conf. Learn. Represent. ICLR 2016 - Conf. Track Proc., Nov. 2015, Accessed: Aug. 02, 2021. [Online]. Available: https:// arxiv. org/ abs/ 1511. 06530v2.

ISSN: 2710-0170

[28] A. Novikov, D. Podoprikhin, A. Osokin, and D. Vetrov, "Tensorizing Neural Networks," Adv. Neural Inf. Process. Syst., vol. 2015-January, pp. 442–450, Sep. 2015, Accessed: Aug. 02, 2021. [Online]. Available: https://arxiv.org/abs/1509.06569v2.

[29] X. Ma et al., "A Tensorized Transformer for Language Modeling," Adv. Neural Inf. Process. Syst., vol. 32, 2019.

[30] K. Ullrich, E. Meeds, and M. Welling, "Soft Weight-Sharing for Neural Network Compression," 5th Int. Conf. Learn. Represent. ICLR 2017 - Conf. Track Proc., Feb. 2017, Accessed: Aug. 02, 2021. [Online]. Available: https://arxiv.org/abs/1702.04008v2.

[31] F. Tung and G. Mori, "Deep Neural Network Compression by In-Parallel Pruning-Quantization," IEEE Trans. Pattern Anal. Mach. Intell., vol. 42, no. 3, pp. 568–579, Mar. 2020, doi: 10.1109/ TPAMI. 2018. 2886192.

[32] A. Polino, R. Pascanu, and D. Alistarh, "Model compression via distillation and quantization," 6th Int. Conf. Learn. Represent. ICLR 2018 - Conf. Track Proc., Feb. 2018, Accessed: Aug. 02, 2021. [Online]. Available: https://arxiv.org/abs/1802.05668v1.

[33] H. Xie, W. Jiang, H. Luo, and H. Yu, "Model compression via pruning and knowledge distillation for person re-identification," J. Ambient Intell. Humaniz. Comput. 2020 122, vol. 12, no. 2, pp. 2149–2161, Jul. 2020, doi: 10.1007/S12652-020-02312-4.

[34] H. Li, A. Kadav, I. Durdanovic, H. Samet, and H. P. Graf, "Pruning Filters for Efficient ConvNets," 5th Int. Conf. Learn. Represent. ICLR 2017 - Conf. Track Proc., Aug. 2016, Accessed: Aug. 02, 2021. [Online]. Available: https://arxiv.org/abs/1608.08710v3.

[35] J. H. Luo, J. Wu, and W. Lin, "ThiNet: A Filter Level Pruning Method for Deep Neural Network Compression," Proc. IEEE Int. Conf. Comput. Vis., vol. 2017-October, pp. 5068–5076, Dec. 2017, doi: 10. 1109/ICCV.2017.541.

[36] M. Elhoushi, Y. H. Tian, Z. Chen, F. Shafiq, and J. Y. Li, "Accelerating Training using Tensor Decomposition," Sep. 2019, Accessed: Aug. 02, 2021. [Online]. Available: https:// arxiv.org/ abs/ 1909. 05675v1.

[37] Z. You, K. Yan, J. Ye, M. Ma, and P. Wang, "Gate Decorator: Global Filter Pruning Method for Accelerating Deep Convolutional Neural Networks," Adv. Neural Inf. Process. Syst., vol. 32, Sep. 2019, Accessed: Sep. 05, 2021. [Online]. Available: https://arxiv.org/abs/1909.08174v1.