LOW-RANK EMBEDDING OF KERNELS IN CONVOLUTIONAL NEURAL NETWORKS UNDER RANDOM SHUFFLING

Chao Li[1,⋆], *Zhun Sun*[1,⋆], *Jinshi Yu*[1,2], *Ming Hou*[1] *and Qibin Zhao*[1,2]

[1]RIKEN Center for Advanced Intelligence Project (AIP), Tokyo 103-0027, Japan
[2]School of Automation, Guangdong University of Technology, Guangzhou 510006, China

ABSTRACT

Although the convolutional neural networks (CNNs) have become popular for various image processing and computer vision tasks recently, it remains a challenging problem to reduce the storage cost of the parameters for resource-limited platforms. In the previous studies, tensor decomposition (TD) has achieved promising compression performance by embedding the kernel of a convolutional layer into a low-rank subspace. However the employment of TD is naively on the kernel or its specified variants. Unlike the conventional approaches, this paper shows that the kernel can be embedded into more general or even random low-rank subspaces. We demonstrate this by compressing the convolutional layers via *randomly-shuffled* tensor decomposition (RsTD) for a standard classification task using CIFAR-10. In addition, we analyze how the spatial similarity of the training data influences the low-rank structure of the kernels. The experimental results show that the CNN can be significantly compressed even if the kernels are randomly shuffled. Furthermore, the RsTD-based method yields more stable classification accuracy than the conventional TD-based methods in a large range of compression ratios.

Index Terms— Deep neural network, weights compression, tensor decomposition, convolutional neural networks

1. INTRODUCTION

Deep convolutional neural networks (CNNs) have advanced to show the state-of-the-art performance in image processing and computer vision applications [1, 2, 3]. However, the huge storage cost of the trainable parameters severely limits its deployment in practice, especially on resource-limited platforms such as smart-phones and wearable devices. For example, the AlexNet Caffemodel is over 200MB and the VGG-16 Caffemodel is over 500MB [4]. Thence, how to efficiently compress the parameters of CNNs has become an urgent task and challenging problem.

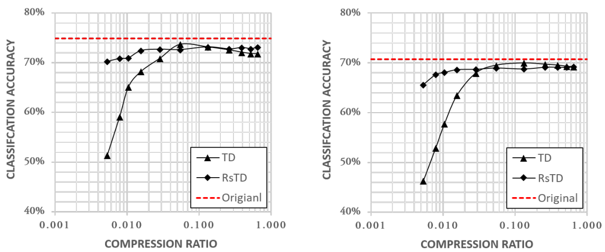
To address this problem, many methods have been proposed, including encoding, quantization and pruning [4]. In recent studies, the deep neural networks are also compressed by the tensor decomposition (TD) models, which embed a multi-way array into lower dimensional spaces [5]. TD itself shows promising results with high compression ratio under less degraded performance [6, 7, 8, 9, 10, 11]. Besides, a combination of TD and aforementioned methods, *e.g.*, encoding and pruning, can further improve the compactness of the CNNs [12]. In early studies, TD was directly employed on the learned kernels. For example, Lebedev *et al.* exploited CANDECOMP/PARAFAC decomposition to compress the convolutional layers in AlexNet [6]. Similarly, Kim *et al.* used Tucker decomposition to speed up various CNNs [11]. More recently, more sophisticated TD models such as tensor train and tensor ring were also employed on compression, in which the kernels are tensorized into higher-order forms for higher compression level [13, 14].

It is worthwhile to mention that, the conventional TD-based compression methods depend on the occurrence of spatial or channel-wised linear dependence within the kernel, thus the kernel can be naturally embedded into a low-rank subspace with lower dimension. Meanwhile, such linear dependence is considered to be occasioned by the spatial similarity of the natural images [15]. Such insight give rise to the following questions: Is the spatial similarity the essential cause of the kernels' low-rank structure? In addition, how does the spatial similarity of the training data influence the low-rank structure the kernels?

To attempt to answer the questions above, we remove the influence of such spatial similarity by randomly shuffling the parameters in the kernels before using TD (see Section 3 for more details). The experimental results reveal an interesting phenomenon that **the TD methods are able to compress CNNs effectively regardless of the random shuffling** (see Figure 1 for example), which implies the fact that the spatial similarity of the training data is not the key factor for embedding the kernels into a low-rank subspace.

Below, we first introduce a unified model for tensor decomposition by using the framework of tensor network. Using this model, we then propose the randomly-shuffled tensor decomposition (RsTD) based convolutional layer, which is used for CNN compression in Section 2. After that, the experiments on two types of CNNs, namely TD-based and

Corresponding Email: {chao.li, qibin.zhao}@riken.jp. ⋆ These authors contributed equally to this work.



(a) CIFAR-10 (b) Noised CIFAR-10

Fig. 1: Comparison of the classification accuracy of the CNNs in our experiments, where TD represents the conventional TD-based compression method (by tensor-train-matrix decomposition), RsTD denotes the proposed model in which the random shuffling operation is imposed on each kernel before TD, and the right line in the figure is the baseline by the uncompressed network.

RsTD-based networks, by using CIFAR-10 dataset follow in Section 3. We conclude with the summary in Section 4.

2. FORMULATION

2.1. A unified model of tensor decomposition

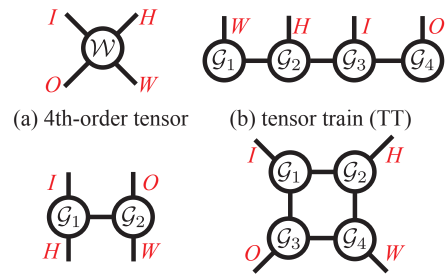
Tensors, also known as multi-way arrays, are generalization of the 2nd-order matrices. Assuming the kernel of a convolutional layer as a 4th-order tensor $\mathcal{W} \in \mathbb{R}^{I\times H\times W\times O}$, where H, W denote the height and width of the filter and I, O denote the number of input and output channels. Then a tensor decomposition model represents it as the product of multiple latent core tensors. Under the framework of tensor network [5], we can mathematically describe TD with a unified model, *i.e.*

$$\mathcal{W} = T_{\mathbf{A}}\left(\mathcal{G}_1, \mathcal{G}_2, \ldots, \mathcal{G}_N\right), \tag{1}$$

where $\mathcal{G}_i, i \in \{1, 2, \ldots, N\}$ denotes the latent core tensors and the operator $T_{\mathbf{A}}$ represents tensor multiplication with given adjacency matrix $\mathbf{A}$ that describe the graph structure of the TD model. For instance, the kernel $\mathcal{W}$ can be decomposed into four cores by tensor train (TT) [16] and the corresponding adjacency matrix is written as

$$\mathbf{A}_{TT} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}, \tag{2}$$

where rows and columns of $\mathbf{A}_{TT}$ correspond different core tensors. We can also describe TD by using graphical representations, in which we consider the vertices as core tensors and the edges as the multiplication of two tensors. Fig. 2 shows three types of TD models including TT, TT-matrix and tensor ring (TR) [17], all of which are used in the experiments



(a) 4th-order tensor (b) tensor train (TT)

(c) TT-matrix (d) tensor ring (TR)

Fig. 2: Graphical representation for decomposing a kernel (4th-order tensor) by using tensor train (TT), TT-matrix and tensor ring (TR) decomposition, respectively.

in this paper. Due to the page limit, more details about TD and its graphical representation can be found from [18] and the references therein.

2.2. Randomly-shuffled TD (RsTD) Layer

The conventional TD-based compression is a method to use the TD models to represent the original kernels in CNN. In this paper, in order to embed the kernel into more diverse low-rank subspaces, we modify the conventional TD-based compression methods by imposing random shuffling operations. Random shuffling is defined as a linear operator R : $\mathbb{R}^{W\times H\times I\times O} \rightarrow \mathbb{R}^{W\times H\times I\times O}$ which *randomly* re-assigns a subscript index for each parameter of the kernel tensor $\mathcal{W}$, such that the parameters of the kernel will be relocated into another place. To simulate the random characteristic by using algorithm, the mapping rule of R can be determined by classical approaches like Fisher-Yates algorithm [19] or its modification [20], which gives equally likely permutation results.

By the random-shuffling operator R, the kernel is decomposed as

$$\tilde{\mathcal{W}} = R \cdot T_{\mathbf{A}}\left(\mathcal{G}_1, \mathcal{G}_2, \ldots, \mathcal{G}_N\right). \tag{3}$$

In contrast to the conventional TD, the kernel generated by (3) will *NOT* have the low-rank structure due to the random shuffling operations. Even though the spatial similarity of the training data yields the low-rank kernels, the newly imposed random shuffling operator is able to invalidate this property. The randomly-shuffled tensor decomposition (RsTD) based convolutional layer is constructed on the basis of (3) with the core tensors $\mathcal{G}_1, \mathcal{G}_2, \ldots, \mathcal{G}_N$ replacing the original kernel $\mathcal{W}$. Mathematically, it can be formulated as

$$\mathcal{Y} = f\left(\left[R \cdot T_{\mathbf{A}}\left(\mathcal{G}_1, \mathcal{G}_2, \ldots, \mathcal{G}_N\right)\right] \otimes \mathcal{X} + \mathcal{B}\right), \tag{4}$$

where the tensor $\mathcal{X}$ and $\mathcal{Y}$ denotes the input and output features of the layer, f denotes the element-wise activation function, $\mathcal{B}$ denotes the bias and $\otimes$ represents the convolution operation[1]. The randomness of R enables the kernels to be embedded into arbitrary low-rank subspaces. Furthermore, note

[1] Here we ignore the stride and padding for brevity.

Input
conv – 3 × 3 – 256 – stride 1 conv – 3 × 3 – 256 – stride 1 conv – 3 × 3 – 256 – stride 2
conv – 3 × 3 – 256 – stride 1 conv – 3 × 3 – 256 – stride 1 conv – 3 × 3 – 256 – stride 2
conv – 3 × 3 – 256 – stride 1 global average pooling fully connected-10
soft-max classifier

Table 1: CNN configurations. The convolution layer parameters are denoted by conv –<kernel size>–<number of output channels>–<stride option>.

that Equation (4) can be degenerated as the conventional TD-based layer if R equals a identical mapping. It implies that the RsTD-based layer is a more general model than the conventional TD-based layer.

3. EXPERIMENTAL RESULTS AND ANALYSIS

3.1. Experiment setting

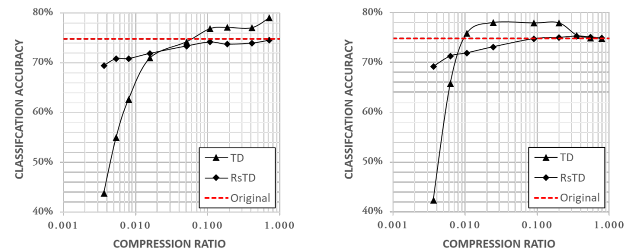
To evaluate the compression capacity of TD and RsTD-based layers, we construct an illustrative CNN by the two types of layers for a standard classification task on the CIFAR-10 dataset [21]. Specifically, we build a prototype CNN with 8 convolutional layers followed by batch normalization [22] and the ReLU activation function. A fully-connected layer is attached at the top of the network, whose outputs are fed into a soft-max classifier. The detailed structure is provided in Table 1. Instead of the conventional CNN, we compress the kernels from the 2nd to the final convolutional layer in the experiment by using TD and RsTD, respectively. Meanwhile, we choose TT-matrix, TT and TR as the decomposition model (see Figure 2), and configure different ranks for each TD model to control the compression level.

During training, we directly update the latent core tensors for each compressed layer using the stochastic gradient decent algorithm with a nesterov moment of 0.9. The initial learning rate is set to be 0.1, decayed by a factor of 10 at epoch 80 and 110, and the training stops at epoch 120. We repeat the training 5 times for each configuration and report the averaged classification accuracy.

Besides the original CIFAR-10 dataset, we further implement 2 variants of CIFAR-10 that are distorted by additional white Gaussian noise (AWGN) of standard deviation 0.4 and 0.8, respectively. Because imposing AWGN on the training data can decrease the spatial similarity from images, we can leverage such the property to analyze how the spatial similarity of the training data influences the low-rank structure of the kernels. In this paper, we use the compression ratio to quantify the compression level for different models. Its formula is given by

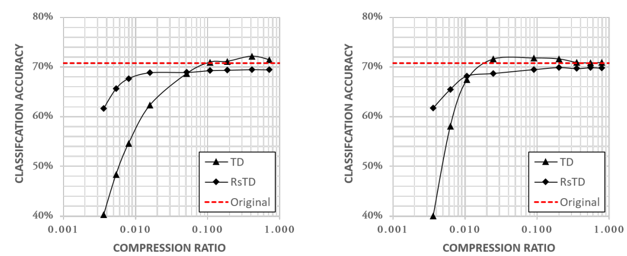
$$r_c = \frac{N_c}{N_u}, \tag{5}$$

where N_c and N_u denote the number of the parameters in the compressed and uncompressed networks, respectively.



(a) TT (b) TR

Fig. 3: Comparison of the test accuracy of CNNs with different compressed layers (TD and RsTD, respectively) by using the original CIFAR-10 dataset. In the figures, the red line denotes the uncompressed baseline, and different sub-figures represent different tensor decomposition models (TT and TR, respectively).

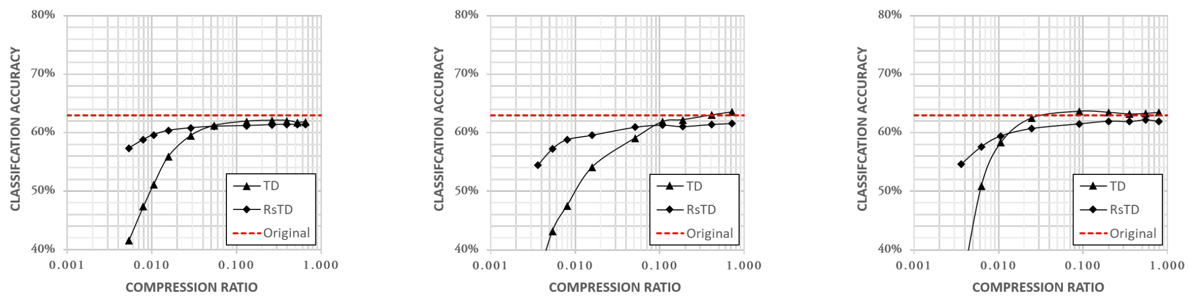


(a) TT (b) TR

Fig. 4: Comparison of the test accuracy of CNNs with different compressed layers (TD and RsTD, respectively) by using the noised CIFAR-10 dataset (dev=0.4). In the figures, the red line denotes the uncompressed baseline, and different sub-figures represent different tensor decomposition models (TT and TR, respectively).

3.2. Results and analysis

We first evaluate the classification performance of the compressed networks by using the original CIFAR-10. Figure 1 (a) and Figure 3 illustrate the classification accuracy of the trained CNNs on different tensor decomposition models with compression ratio ranging from 0.003 to 1. As shown in the figures, both TD and RsTD-based layers obtain similar classification accuracy compared to the baseline (uncompressed networks, the red line in the figures) when the compression



(a) TT-matrix

(b) TT

(c) TR

Fig. 5: Comparison of the test accuracy of CNNs with different compressed layers (TD and RsTD, respectively) by using the noised CIFAR-10 dataset (dev=0.8). In the figures, the red line denotes the uncompressed baseline, and different sub-figures represent different tensor decomposition models (TT and TR, respectively).

ratio is relatively high ($r_c > 0.05$). It implies that the storage overhead of the compressed CNN is 20 times smaller than the uncompressed counterpart by using both TD and RsTD-based layers without significant performance loss. The accuracy of RsTD is competitive with that of baseline, which reflects the fact that the shuffled kernels can be still embedded into low-rank subspaces, even if the mapping rule R is randomly chosen.

The performance curves in these figures render two interesting phenomena. First, the accuracy of the TD-based layers go down dramatically as the compression ratio decreases, whereas the RsTD-based layers is able to maintain relatively high classification accuracy. The inferior performance of the TD-based layers is due to the under-fitting problem of the network. For example, assume that we decompose the kernel by using rank-1 TR decomposition ($r_c \approx 0.004$), then Eq. (4) can be rewritten as

$$= f\left(\mathcal{G}_3 \times_{1,3} \left(merge\left(\mathcal{G}_1, \mathcal{G}_2, \mathcal{G}_4\right) \otimes \mathcal{X}\right) + \mathcal{B}\right), \qquad (6)$$

where the operators $\times_{1,3}$ and $merge(\cdot)$ denote the multiplication and merging operation of the core tensors [14], respectively. We can see from (6) that the operations in the convolutional layer can be split into two steps. In the first step, the input feature $\mathcal{X}$ is convolved with the merged cores $\mathcal{G}_i, i = 1, 2, 4$, which is equivalent to mapping $\mathcal{X}$ into a latent space *w.r.t.* the outputs. Hence, the rank-1 TR model implies that the features for all output channels are identical to each other up to scale, which naturally leads to under-fitting problem in most of CNN learning models. In contrast, as mentioned in Section 2, imposing random shuffling can increase the rank of the kernel, which suggests to increase the dimension of the latent space. Such property can actually mitigate the under-fitting issue of the network.

The second phenomenon is that TT and TR-based layers outperform both the baseline and RsTD-based layers in the case of high compression ratio. This observation supports the conventional claim that the spatial similarity of the training data results in the linear dependence within the kernel. To find out the reason behind the performance gap between TD and RsTD-based layers, we validate our model on the noisy CIFAR-10 (dev $= 0.4$ and 0.8, respectively) to train the networks. Figure 1 (b) and Figure 4 give the classification accuracy when dev $= 0.4$, and Figure 5 illustrate the result when dev $= 0.8$. As depicted in the figures, the performance gap between TD and RsTD becomes smaller as the noise strength increases. This is because imposing the noise will decrease the spatial similarity of the training data. In these cases, the RsTD-based layer shows more reliable performance with a larger range of compression ratios.

4. CONCLUSION

In this paper, we consider embedding the kernels in CNN into more general low-rank subspaces and analyze the impact of spatial similarity of the training data on the low-rank structure of the kernel. For this purpose, we impose the random shuffling operations before tensor decomposition. The experimental results demonstrate that RsTD can be exploited to compress the kernels in CNN without significant performance loss. Also, CNNs equipped with RsTD-based layers overwhelmingly outperform those with un-shuffled kernels under significantly small compression ratio. Furthermore, decreasing the spatial similarity of the training data diminishes the diversity of performance between TD and RsTD based models. The conclusions can be made from the experimental results that the kernels in CNN have an inherent low-rank structure regardless of the structure of the training data.

Acknowledgement

This work was supported by JSPS KAKENHI (Grant No. 17K00326), JST CREST (Grant No. JPMJCR1784), and the National Natural Science Foundation of China (Grant No. 61773129).

5. REFERENCES

[1] Dmitry Ulyanov, Andrea Vedaldi, and Victor Lempitsky, "Deep image prior," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, vol. 18.

[2] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," in *Advances in neural information processing systems*, 2015, pp. 91–99.

[3] Chao Dong, Chen Change Loy, Kaiming He, and Xiaoou Tang, "Learning a deep convolutional network for image super-resolution," in *European conference on computer vision*. Springer, 2014, pp. 184–199.

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

[5] Andrzej Cichocki, Danilo Mandic, Lieven De Lathauwer, Guoxu Zhou, Qibin Zhao, Cesar Caiafa, and Huy Anh Phan, "Tensor decompositions for signal processing applications: From two-way to multiway component analysis," *IEEE Signal Processing Magazine*, vol. 32, no. 2, pp. 145–163, 2015.

[6] Vadim Lebedev, Yaroslav Ganin, Maksim Rakhuba, Ivan Oseledets, and Victor Lempitsky, "Speeding-up convolutional neural networks using fine-tuned cp-decomposition," *arXiv preprint arXiv:1412.6553*, 2014.

[7] Alexander Novikov, Dmitrii Podoprikhin, Anton Osokin, and Dmitry P Vetrov, "Tensorizing neural networks," in *Advances in Neural Information Processing Systems*, 2015, pp. 442–450.

[8] Edwin Stoudenmire and David J Schwab, "Supervised learning with tensor networks," in *Advances in Neural Information Processing Systems*, 2016, pp. 4799–4807.

[9] Jean Kossaifi, Zachary C Lipton, Aran Khanna, Tommaso Furlanello, and Anima Anandkumar, "Tensor regression networks," *arXiv preprint arXiv:1707.08308*, 2017.

[10] Yongxin Yang and Timothy Hospedales, "Deep multi-task representation learning: A tensor factorisation approach," *arXiv preprint arXiv:1605.06391*, 2016.

[11] Yong-Deok Kim, Eunhyeok Park, Sungjoo Yoo, Taelim Choi, Lu Yang, and Dongjun Shin, "Compression of deep convolutional neural networks for fast and low power mobile applications," *arXiv preprint arXiv:1511.06530*, 2015.

[12] Anonymous, "Exploiting invariant structures for compression in neural networks," in *Submitted to International Conference on Learning Representations*, 2019, under review.

[13] Timur Garipov, Dmitry Podoprikhin, Alexander Novikov, and Dmitry Vetrov, "Ultimate tensorization: compressing convolutional and fc layers alike," *arXiv preprint arXiv:1611.03214*, 2016.

[14] Wenqi Wang, Yifan Sun, Brian Eriksson, Wenlin Wang, and Vaneet Aggarwal, "Wide compression: Tensor ring nets," *learning*, vol. 14, no. 15, pp. 13–31, 2018.

[15] Misha Denil, Babak Shakibi, Laurent Dinh, Nando De Freitas, et al., "Predicting parameters in deep learning," in *Advances in neural information processing systems*, 2013, pp. 2148–2156.

[16] Ivan V Oseledets, "Tensor-train decomposition," *SIAM Journal on Scientific Computing*, vol. 33, no. 5, pp. 2295–2317, 2011.

[17] Qibin Zhao, Guoxu Zhou, Shengli Xie, Liqing Zhang, and Andrzej Cichocki, "Tensor ring decomposition," *arXiv preprint arXiv:1606.05535*, 2016.

[18] Andrzej Cichocki, Namgil Lee, Ivan Oseledets, Anh-Huy Phan, Qibin Zhao, Danilo P Mandic, et al., "Tensor networks for dimensionality reduction and large-scale optimization: Part 1 low-rank tensor decompositions," *Foundations and Trends® in Machine Learning*, vol. 9, no. 4-5, pp. 249–429, 2016.

[19] Vladimir Batagelj and Ulrik Brandes, "Efficient generation of large random networks," *Physical Review E*, vol. 71, no. 3, pp. 036113, 2005.

[20] Richard Durstenfeld, "Algorithm 235: random permutation," *Communications of the ACM*, vol. 7, no. 7, pp. 420, 1964.

[21] Alex Krizhevsky and Geoffrey Hinton, "Learning multiple layers of features from tiny images," Tech. Rep., Citeseer, 2009.

[22] Sergey Ioffe and Christian Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," *arXiv preprint arXiv:1502.03167*, 2015.