CEPD: CO-EXPLORING PRUNING AND DECOMPOSITION FOR COMPACT DNN MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

Pruning and decomposition are two important techniques to compress deep neural network (DNN) models. To date, these two popular yet distinct approaches are typically used in a separate way; while their efficient integration for better compression performance is little explored. In this paper, we perform systematic co-exploration on pruning and decomposition toward compact DNN models. We first investigate and analyze several important design factors for joint pruning and decomposition, including operational sequence, decomposition format, and optimization procedure. Based on the observations from our analysis, we then propose CEPD, a unified DNN compression framework that can simultaneously capture the benefits of pruning and decomposition in an efficient way. Empirical experiments demonstrate the promising performance of our proposed solution. Notably, on CIFAR-10 dataset, CEPD brings 0.72% and 0.45% accuracy increase over the baseline ResNet-56 and MobileNetV2 models, respectively, and meanwhile the computational costs are reduced by 43.0% and 44.2%, respectively. On the ImageNet dataset, our approach can enable 0.10% and 1.39% accuracy increase over the baseline ResNet-18 and ResNet-50 models with 59.4% and 54.6% fewer parameters, respectively.

1 INTRODUCTION

Deep neural network (DNN) has served as the backbone machine learning technique in many modern intelligent systems. To facilitate the low-cost deployment of DNN on the resource-constrained platforms, *model compression*, as a powerful strategy that can efficiently reduce DNN model size, has been extensively studied in recent years. Among various types of model compression techniques, *sparsification (a.k.a., pruning)* and *low-rank decomposition* are two representative and popular solutions (Wang et al. (2021); Gao et al. (2021); Li et al. (2021a); Xu et al. (2020)). As revealed by their names, the low-rank and sparse methods aim to explore and leverage the potential low-rankness and sparsity of the uncompressed DNNs, respectively.

Co-exploring Low-rankness & Sparsity: Motivation. Considering the current prosperity of these two methods and their very distinct structural assumptions, an interesting and promising research topic is to explore the efficient integration of low-rank and sparse approaches towards a better model compression solution. As indicated and observed by (Yu et al. (2017)), DNN models tend to exhibit both low-rankness and sparsity simultaneously. For instance, the smooth components in the weight filters can be represented in the low-rank space, and meanwhile some other important information is sparsely scattered. Evidently, fully leveraging such co-existence of these structure-level patterns can potentially bring a powerful compression solution with attractive performance.

Existing works. Unlike the current extensive research activities on individual low-rank and sparse methods, the investigations on integrating these two approaches, in an efficient and non-trivial way, are little explored. To date, only very few efforts study the joint exploration of low-rankness and sparsity for DNN model compression. As the pioneering work in this direction, (Yu et al. (2017)) develops a singular value decomposition (SVD)-free approach to closely approximate original DNN model via combining sparse representation and low-rank matrix factorization. Built on the interesting connection between filter decomposition and filter pruning, (Li et al. (2020)) interprets the decomposition and pruning of convolutional filters in a unified perspective. (Idelbayev & Carreira-Perpinán (2021)) combines low-rank, sparse and quantized matrices into an additive framework with the learning-compression algorithm. Most recently, (Li et al. (2021b)) proposes a collaborative

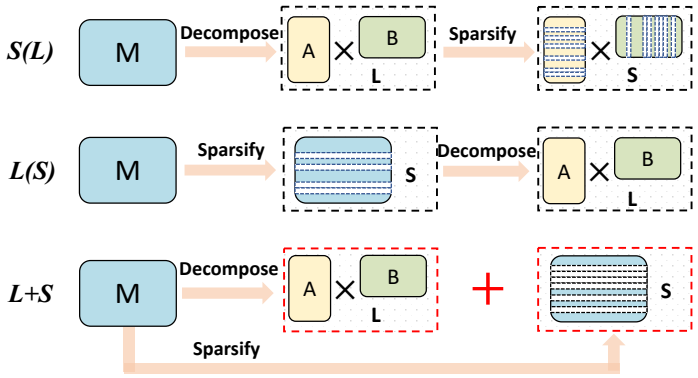

Figure 1: Different operational sequences for joint low-rank decomposition and **structured** pruning. Here ***L*** and ***S*** represent low-rank decomposition and sparsification, respectively.

compression scheme to integrate SVD into model sparsification. By adopting a multi-step heuristic removal strategy, this post-training approach achieves promising task and compression performance.

Unanswered Questions. Although these prior works have demonstrated the huge potential and attractive benefits of jointly decomposing and pruning, the systematic investigation of their efficient integration is still missing. To be specific, several fundamental and critical questions, whose answers will directly impact the integration scheme and overall compression performance, have not been comprehensively explored yet. For instance, because pruning and decomposition can be jointly performed in several different ways, such as in parallel (Yu et al. (2017); Idelbayev & Carreira-Perpinán (2021)) or in sequence (Li et al. (2021b)), which collaborative strategy is more suitable for the target DNN compression task? Also, considering low-rankness can be exploited from different perspectives, which type of low-rank approach should be adopted? The matrix factorization used in (Li et al. (2020; 2021b))? Or even high-order tensor decomposition? In addition, to achieve promising compression performance, what is the most suitable optimization objective that the integration scheme should aim for? The approximation error focused in (Yu et al. (2017))? The low-rankness/sparsity regularized loss in (Yang et al. (2020))? Or some other new alternatives?

Technical Preview and Contributions. To answer these questions and develop an efficient integrated model compression solution, in this paper we perform systematic co-exploration on the model low-rankness and sparsity towards compact neural networks. To be specific, we first review and analyze several important design factors for the joint low-rank decomposition and *structured* pruning. Based on the observations and outcomes from our analysis, we then propose CEPD, a unified DNN compression framework that can simultaneously capture model low-rankness and structured sparsity in an efficient way. Overall, the contributions of this paper are summarized as follows:

- We systematically investigate and analyze the critical design knobs when co-exploring model low-rankness and sparsity, including operational sequence, low-rank format, and optimization objective. Based on our qualitative and quantitative analysis, we propose several recommended design options for efficient joint low-rank decomposition and pruning.
- We develop a unified framework that formulates the integration of low-rank decomposition and pruning to an optimization problem with low-tensor-rank and sparse constraints. We then derive a training-aware approach to solve this challenging non-convex high-order tensor-format problem, and thereby leading to efficient exploration of rich low-rankness and sparsity in the model.
- We empirically evaluate our proposed co-exploration solution for various DNN models on different datasets, and the experimental results demonstrate its promising performance.

2 Related Work

Pruning. Network pruning, which explores the sparse property, has been extensively studied for model compression. Existing pruning methods can be performed with different granularity.

Unstrcutured pruning prunes weights individually based on a certain criterion (Han et al. (2015)). It usually brings outstanding performance but requires complicated hardware design or dedicated sparse matrix operation library to be deployed in practice. Structured pruning prunes weights in groups, either removing the entire convolutional filters or feature map channels (Liebenwein et al. (2019); Lin et al. (2020); Tang et al. (2020); Sui et al. (2021); Ganjdanesh et al. (2022)). In this paper, we focus on co-exploring low rankness and the sparsity brought by structured pruning.

Low-rank Decomposition. Low-rank decomposition is another popular DNN compression approach that captures the low-rankness of the model. Based on different interpretation of DNNs, this method can be categorized to *matrix decomposition* and *tensor decomposition*. Matrix decomposition views the 4-D weight tensor as the folded matrix, and hence it flattens the 4-D objective to 2-D format and decomposes the reshaped matrix to the product of two small matrices (Tai et al. (2016); Li & Shi (2018); Xu et al. (2020); Idelbayev & Carreira-Perpinán (2020); Liebenwein et al. (2021)). However, the existing matrix decomposition-based works suffer the loss of important spatial information incurred by inevitable tensor flatten operations. On the other aspect, tensor decomposition directly factorizes the 4-D weight tensor to multiple small tensor cores without flatten operations. Such explicit high-order processing, by its nature, can better preserve the important spatial information and correlation that existed in the weight tensors. To date several tensor decomposition techniques, such as Tucker/CP, tensor train, and tensor ring etc., have been used for DNN model compression (Denton et al. (2014); Kim et al. (2016); Novikov et al. (2015); Wang et al. (2018); Kossaifi et al. (2019; 2020); Phan et al. (2020); Li et al. (2021a)).

Joint Pruning and Decomposition. As observed by (Yu et al. (2017)), a well-trained DNN tends to exhibit both sparsity and low-rankness simultaneously. Motivated by this observation, some prior efforts propose to co-explore these two complementary properties for model compression. As the pioneering work, (Yu et al. (2017)) decomposes the weight tensors of a pre-trained DNN model into independent low-rank and sparse parts and minimizes the reconstruction error. Similarly, (Ma et al. (2019)) focus on optimizing the approximation error of the original model via ADMM. Different from this parallel scheme, (Dubey et al. (2018); Li et al. (2021b)) adopt a sequential compression strategy via performing matrix factorization on a pruned model. In addition, (Li et al. (2020)) proposes to use the sparse/low-rank regularization term, instead of reconstruction error, to enforce the desired structural patterns. (Idelbayev & Carreira-Perpinán (2021)) proposes a general additive combination framework with learning-compression algorithm. Also, notice that all of the existing works focus on using either SVD-based or SVD-free matrix decomposition to exploit the low-rankness of DNN model. However, as analyzed in Section 3 of our paper, high-order tensor decomposition is a more suitable low-rank methods for CNN compression; while this important technique has not been explored by the existing additive compression efforts yet. Besides, they adopt either approximation-centered or regularization-based optimization methods. We believe such decomposition and optimization solutions have limitations and cannot achieve optimal performance, since the task goal is to generate a compressed model instead of close approximation to the original model. Different from these existing works, our approach is built on tensor decomposition with constrained formulation as the optimization objective, demonstrating better performance.

3 Co-exploring Low-rankness and Sparsity: Analysis

As outlined above, some prior works have reported their exploration of joint low-rankness and sparsity. In general, the integration of low-rank decomposition and pruning can be specified by several important factors, including operational sequence, low-rankness format and overall optimization objective. The existence of such a large variety of different factors and their combinations, by its nature, calls for the systematic investigation of the best-suited co-exploration scheme. Such an analysis framework, if being properly developed, can facilitate the optimal selection of various design factors already proposed in the existing literature. More importantly, the outcome of this systematic study will further guide and provide better integration choices that have not been discovered before.

Questions to be Answered. Next we analyze the critical design knobs for efficient co-exploration on model low-rankness and sparsity. To that end, three important questions need to be answered.

Question #1: *What is the more suitable operational sequence when jointly low-rank decomposing and pruning DNN models?*

Analysis. In general, the co-existence of model low-rankness and sparsity can be explored in different ways (see Figure 1). For instance, as adopted in (Yu et al. (2017)), a well-trained DNN can be closely approximated as the combination of a low-rank component and a sparse component. In other words, the two types of structure-level properties are imposed and leveraged in a *spatially parallel way*, and we denote this strategy as ***L+S***, where ***L*** and ***S*** represent low-rank decomposition and sparsification, respectively. On the other hand, the joint use of factorization and pruning can also be performed in a *temporally sequential way*. As illustrated in Figure 1, the original model can be first imposed with low-rankness (or sparsity), and the size of the resulting partially compressed model can be further reduced by the second-stage pruning (or low-rank decomposition). Following the similar notation, such sequential operation can be denoted as ***S(L)*** and ***L(S)***. In practice ***L(S)*** is a preferable choice that has been adopted in the prior works (Dubey et al. (2018); Li et al. (2021b)).

Our Proposal. Among the above described three general operational schemes, we believe ***L+S*** is the more suitable choice when considering to integrate pruning and decomposition together for model compression. This is because unlike ***S(L)*** and ***L(S)***, which ultimately still produce the compressed model in a single representation (sparse or low-rank) space, ***L+S*** enables the simultaneous representation of rich information of DNN models across different subspace, and thereby better preserving the structural characteristics and reducing the potential information loss. To verify our hypothesis, we examine the approximation error incurred by three integration schemes. As shown in Figure 2 and more detailed results of **Appendix** D.1, with the same compression ratio for the weight tensor of one layer of a pre-trained ResNet-20 on CIFAR-10 dataset, ***L+S*** shows lower approximation error than its counterparts. This experimental phenomenon demonstrates that ***L+S*** scheme indeed can capture both the low-rank and sparse characteristics of DNN model in an efficient way.

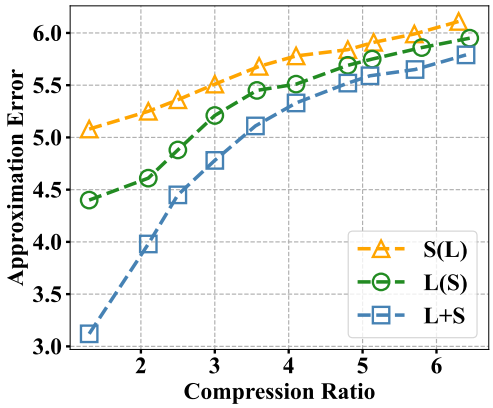

Figure 2: The approximation error when compressing the weight tensor of one layer in ResNet-20 using different operational sequences (***L+S***, ***S(L)*** and ***L(S)***). Here mean square error (MSE) is used to measure the difference between the original uncompressed weight tensor and the reconstruction. SVD is adopted as the low-rank decomposition method (***L***). It is seen that ***L+S*** can bring much smaller approximation error than its counterparts with the same compression ratio. More comprehensive layer-wise results of ResNet-20 and ResNet-56 on CIFAR-10 and ResNet-50 on ImageNet are reported in **Appendix** D.1.

Question #2: *What is the more suitable low-rank decomposition approach used when co-exploring low-rankness and sparsity?*

Analysis. From the perspective of linear algebra, the low-rankness of a DNN model can be exploited using different ways. For an example convolutional layer, imposing the low-rank structure can be realized by performing simple matrix factorization or high-order tensor decomposition (see Figure 4 in **Appendix** A). Specifically, (Yu et al. (2017)) chooses SVD-free method to factorize DNN model and obtain the low-rank component, and (Li et al. (2021b)) proposes to use SVD-based decomposition to serve as the second-stage compression approach in its adopted ***L(S)*** scheme. Similarly, the low-rank methods adopted in (Ma et al. (2019); Idelbayev & Carreira-Perpinán (2021)) are also based on 2-D matrix. Notice that though the weights of convolutional layer essentially form a 4-D tensor format, the existing works exploit the low-rankness via using matrix decomposition – the 4-D tensor needs to be first flattened to a 2-D matrix and it is then factorized to two small matrices.

Our Proposal. We argue that the high-order tensor decomposition, the option that has not been explored in the integration scheme before, is the better choice than the low-order matrix decomposition adopted in the existing works. This is because as a reshaping-free technique that can directly factorize the tensor-format data to multiple tensor cores, tensor decomposition, such as Tensor Train (TT) and Tucker, can naturally capture and preserve the important spatial information and correlation of the original weight tensors in a more efficient way. Therefore, less information loss is expected after performing low-rank tensor-based compression. To verify our hypothesis, we compare the feature

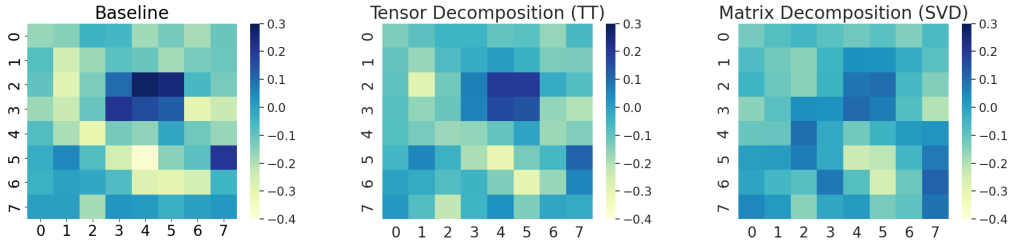

Figure 3: Output feature maps of one layer of ResNet-20 after non-compression (Left), tensor decomposition (Middle) and matrix decomposition (Right). The visualization shown here is based on the information of one channel. **It is seen that high-order tensor decomposition makes the feature map of the compressed layer more similar to that of the original uncompressed layer.**

maps of the compressed convolutional layer of ResNet-20 on CIFAR-10 dataset using different low-rank methods. For SVD, we follow the same way used in (Li et al. (2021b)) to reshape the 4-D tensor weight to a 2-D weight matrix. As visualized in Figure 3, compared with the matrix decomposition-based approach with the same compression ratio, tensor decomposition can make the output feature map of the compressed layer much more similar to the feature map of the original uncompressed layer. In other words, low-rank tensor method can provide better preservation of the important feature information and thus it can bring potential higher model compression performance. More detailed quantitative comparison results for different low-rank methods (SVD, Tucker and TT) under different compression ratio are reported in **Appendix** D.2.

Question #3: *What is the suitable optimization objective that the joint compression should aim for?*

Analysis. To efficiently realize the joint exploration of model low-rankness and sparsity with promising compression performance, different optimization strategies have been proposed in the existing works. For instance, (Yu et al. (2017); Ma et al. (2019)) aim to minimize the difference between the original weight matrix/tensor and the approximated reconstruction. This type of solution essentially applies the general sparsity/low-rankness co-exploration methods (Bouwmans et al. (2016); Richard et al. (2013; 2012); Luo (2011)) in linear algebra and signal processing fields to DNN compression. However, such matrix recovery/estimation-oriented strategy is not the optimal solution for DNN model compression, since our task goal is to generate a compressed model instead of close approximation to the original model. In addition, (Xu et al. (2020); Yang et al. (2020)) explore another strategy that they consider global information via adding regularization term to the loss function. Besides, (Ma et al. (2019); Guo et al. (2019)) also propose to explicitly add the low-rank and sparse regularization terms to the overall objective function, which can guide the training-aware procedure to enforce the desired structural patterns.

Our Proposal. Different from the existing approximation error-centered or regularized loss-based solutions, we propose that the efficient co-exploration scheme should be interpreted as the optimization procedure with the low-rank and sparse constraints. Our rationale lies on two observations of the drawbacks of the prior efforts. First, the approximation strategy adopted in (Yu et al. (2017); Ma et al. (2019)) focuses on making the reconstructed model approach the original model as close as possible. However, since 1) the approximation error always exists; and 2) the original model is not the only choice to achieve the desired accuracy, such a strategy inherently can only search the low-rank and sparse components in a limited exploration space, thereby affecting the overall performance.

Second, though adding the regularization terms into the loss function indeed facilitates the extraction of low-rank and sparse patterns, the effect of such a simple regularizing method can only approximately satisfy the hard constraint, which is still limited, especially considering the efforts of pushing for sparsity and for low-rankness may interfere with each other, thereby potentially causing unexpected conflicts. Instead, by explicitly imposing the low-rank and sparse constraints on the overall optimization problem, these two structural requirements can be simultaneously satisfied with the proper use of optimization technique (to be discussed in Section 4). To be specific, we report the learning curve in **Appendix** B to show that our proposed constrained optimization strategy can successfully impose the desired low-rankness and sparsity onto the DNN models efficiently. We also provide a quantitative comparison in Section 5 to show the better performance of the proposed method over approximation-centered and regularized loss-based solutions.

Summary of Our Analysis. ❶ Performing joint low-rank decomposition and structured pruning in a spatially parallel way (***L+S***) is the preferable operational sequence. ❷ High-order tensor decomposition is the more suitable choice for the low-rank approach used in the integrated compression scheme. ❸ Imposing low-rankness and sparsity as the direct hard constraints on the loss optimization should be adopted to better satisfy the desired structural requirement.

4 CO-EXPLORING LOW-RANKNESS AND SPARSITY: OUR METHOD

Problem Formulation. Recall that the analysis in Section 3 brings three important observations/proposals: using ***L+S*** operational sequence, choosing high-order tensor decomposition, and directly imposing hard constraints. Built on such three fundamental principles, we are now ready to formulate the integration of pruning and tensor decomposition to a unified optimization problem. To be specific, given an uncompressed DNN model with weight tensor $\boldsymbol{\mathcal{W}} \in \mathbb{R}^{O\times I\times K\times K}$ of each layer, our goal is to find another compact model with weight tensors $\boldsymbol{\mathcal{L}} + \boldsymbol{\mathcal{S}}$, which consists of low-rank component $\boldsymbol{\mathcal{L}} \in \mathbb{R}^{O\times I\times K\times K}$ and structured sparse component $\boldsymbol{\mathcal{S}} \in \mathbb{R}^{O'\times I'\times K\times K}$ for each layer, to minimize the following loss function:

$$\min_{\boldsymbol{\mathcal{L}},\boldsymbol{\mathcal{S}}} f(\boldsymbol{\mathcal{L}},\boldsymbol{\mathcal{S}}),\ \text{ s.t. } \underbrace{r(\boldsymbol{\mathcal{L}}) \le \gamma_0,\gamma_1,\cdots,\gamma_d,}_{\text{Low-tensor-rank constraint}}\ \underbrace{c(\boldsymbol{\mathcal{S}}) \le \kappa}_{\text{Sparse constraint}}\ , \tag{1}$$

where $f(\cdot)$ is the loss over the entire training dataset, $r(\cdot)$ calculates the rank of a tensor , $c(\cdot)$ is the number of unpruned filters, and $\gamma_0,\gamma_1,\cdots,\gamma_d$ and κ are the desired tensor ranks and the number of remaining filters for $\boldsymbol{\mathcal{L}}$ and $\boldsymbol{\mathcal{S}}$, respectively. Without loss of generality, we choose tensor train (TT) decomposition as the component low-rank method and ℓ_1 norm as the criterion to prune a filter in our framework. Here d is the number of decomposed tensor cores with TT decomposition.

Method. Directly optimizing problem (1) is challenging because of the co-existence of the non-differentiable r$(\cdot)$ and c$(\cdot)$ as well as its inherent high-order tensor format. To efficiently solve this problem, we propose to leverage alternating direction optimization method to split these two constraints. To be specific, after introducing two auxiliary variables $\widehat{\boldsymbol{\mathcal{L}}}$ and $\widehat{\boldsymbol{\mathcal{S}}}$ that represent the desired low-TT-rankness and structured sparsity in the optimization process, problem (1) can be then rewritten as:

$$\min_{\boldsymbol{\mathcal{L}},\boldsymbol{\mathcal{S}},\widehat{\boldsymbol{\mathcal{L}}}\in\mathcal{P},\widehat{\boldsymbol{\mathcal{S}}}\in\mathcal{Q}} f(\boldsymbol{\mathcal{L}},\boldsymbol{\mathcal{S}}),\ \text{ s.t. } \boldsymbol{\mathcal{L}}=\widehat{\boldsymbol{\mathcal{L}}},\boldsymbol{\mathcal{S}}=\widehat{\boldsymbol{\mathcal{S}}}, \tag{2}$$

where $\mathcal{P}=\{\boldsymbol{\mathcal{L}}|r(\boldsymbol{\mathcal{L}})\le\gamma_1,\cdots,\gamma_d\}$ is the set of all tensors that satisfy the low-tensor-rank constraint, and $\mathcal{Q}=\{\boldsymbol{\mathcal{S}}|c(\boldsymbol{\mathcal{S}})\le\kappa\}$ is the set of all tensors that satisfy the structured sparse constraint. Then, we further relax the hard constraints to the corresponding augmented Lagrangian form and now we only need to optimize the following new constraint-free min-max problem:

$$\min_{\boldsymbol{\mathcal{L}},\boldsymbol{\mathcal{S}},\widehat{\boldsymbol{\mathcal{L}}}\in\mathcal{P},\widehat{\boldsymbol{\mathcal{S}}}\in\mathcal{Q}}\ \max_{\boldsymbol{\mathcal{U}},\boldsymbol{\mathcal{V}}}\ f(\boldsymbol{\mathcal{L}},\boldsymbol{\mathcal{S}})+\frac{\lambda}{2}(\|\boldsymbol{\mathcal{L}}-\widehat{\boldsymbol{\mathcal{L}}}+\boldsymbol{\mathcal{U}}\|_F^2+\|\boldsymbol{\mathcal{S}}-\widehat{\boldsymbol{\mathcal{S}}}+\boldsymbol{\mathcal{V}}\|_F^2+\|\boldsymbol{\mathcal{U}}\|_F^2+\|\boldsymbol{\mathcal{V}}\|_F^2), \tag{3}$$

where $\boldsymbol{\mathcal{U}}$ and $\boldsymbol{\mathcal{V}}$ are the dual multipliers associated to $\boldsymbol{\mathcal{L}}$ and $\boldsymbol{\mathcal{S}}$, respectively, and λ is the penalty parameters. To solve this minmax problem, we can split it into three separated parts, and independently optimize them in an iterative way.

Update $\boldsymbol{\mathcal{L}}$ and $\boldsymbol{\mathcal{S}}$ with SGD. The first independent optimization objective can be formulated as:

$$\min_{\boldsymbol{\mathcal{L}},\boldsymbol{\mathcal{S}}}\ f(\boldsymbol{\mathcal{L}},\boldsymbol{\mathcal{S}})+\frac{\lambda}{2}(\|\boldsymbol{\mathcal{L}}-\widehat{\boldsymbol{\mathcal{L}}}+\boldsymbol{\mathcal{U}}\|_F^2+\|\boldsymbol{\mathcal{S}}-\widehat{\boldsymbol{\mathcal{S}}}+\boldsymbol{\mathcal{V}}\|_F^2). \tag{4}$$

Since there are no hard constraints on the target variables $\boldsymbol{\mathcal{L}}$ and $\boldsymbol{\mathcal{S}}$, standard DNN optimizer (e.g., stochastic gradient descent (SGD)) can be directly applied with learning rate α as:

$$\boldsymbol{\mathcal{L}} \leftarrow \boldsymbol{\mathcal{L}}-\alpha[\nabla_{\boldsymbol{\mathcal{L}}} f(\boldsymbol{\mathcal{L}},\boldsymbol{\mathcal{S}})+\lambda(\boldsymbol{\mathcal{L}}-\widehat{\boldsymbol{\mathcal{L}}}+\boldsymbol{\mathcal{U}})], \tag{5}$$

$$\boldsymbol{\mathcal{S}} \leftarrow \boldsymbol{\mathcal{S}}-\alpha[\nabla_{\boldsymbol{\mathcal{S}}} f(\boldsymbol{\mathcal{L}},\boldsymbol{\mathcal{S}})+\lambda(\boldsymbol{\mathcal{S}}-\widehat{\boldsymbol{\mathcal{S}}}+\boldsymbol{\mathcal{V}})]. \tag{6}$$

Update $\widehat{\boldsymbol{\mathcal{L}}}$ with TT Decomposition. To update the introduced $\widehat{\boldsymbol{\mathcal{L}}}$, the optimization objective is:

$$\min_{\widehat{\boldsymbol{\mathcal{L}}}\in\mathcal{P}}\ \frac{\lambda}{2}\|\boldsymbol{\mathcal{L}}-\widehat{\boldsymbol{\mathcal{L}}}+\boldsymbol{\mathcal{U}}\|_F^2. \tag{7}$$

Because $\widehat{\boldsymbol{\mathcal{L}}}$ is strictly constrained to stay in the low-tensor-rank set $\mathcal{P}$, the desired update can be performed using an analytical solution via TT-rank truncation, i.e.

$$\widehat{\boldsymbol{\mathcal{L}}} \leftarrow \mathbf{trunc}_{\mathcal{P}}(\boldsymbol{\mathcal{L}} + \boldsymbol{\mathcal{U}}). \tag{8}$$

To realize the truncating operation, we first define a temporary tensor $\boldsymbol{\mathcal{T}} = \boldsymbol{\mathcal{L}} + \boldsymbol{\mathcal{U}}$ and reshape it as a new tensor $\widetilde{\boldsymbol{\mathcal{T}}} \in \mathbb{R}^{(K\times K)\times(O_1\times I_1)\times\cdots\times(O_d\times I_d)}$ with $O = \prod_{k=1}^{d} O_k$, $I = \prod_{k=1}^{d} I_k$. Then $\widetilde{\boldsymbol{\mathcal{T}}}$ can be decomposed to $d+1$ TT-cores as:

$$\widetilde{\boldsymbol{\mathcal{T}}}((k_1,k_2),(o_i,i_1),\cdots,(o_d,i_d))) = \boldsymbol{\mathcal{C}}_0(k_1,k_2)\boldsymbol{\mathcal{C}}_1(:,o_1,i_1,:)\cdots\boldsymbol{\mathcal{C}}_d(:,o_d,i_d,:), \tag{9}$$

where $\boldsymbol{\mathcal{C}}_0 \in \mathbb{R}^{K\times K}, \boldsymbol{\mathcal{C}}_j \in \mathbb{R}^{R_{j-1}\times O_j\times I_j\times R_j}, j = 1,\cdots,d$. In this TT-format, the dimensions of TT-ranks in TT-cores are truncated to the desired target, i.e., $\boldsymbol{\mathcal{C}}'_j = \boldsymbol{\mathcal{C}}_j(1:\gamma_{j-1},:,:,1:\gamma_j)$. After that we use the truncated TT-cores to recover the original tensor via:

$$\widetilde{\boldsymbol{\mathcal{T}}}'((k_1,k_2),(o_i,i_1),\cdots,(o_d,i_d))) = \boldsymbol{\mathcal{C}}_0(k_1,k_2)\boldsymbol{\mathcal{C}}'_1(:,o_1,i_1,:)\cdots\boldsymbol{\mathcal{C}}'_d(:,o_d,i_d,:). \tag{10}$$

And finally $\widetilde{\boldsymbol{\mathcal{T}}}'$ is reshaped to the original shape of $\widehat{\boldsymbol{\mathcal{L}}}$ to serve as the updated $\widehat{\boldsymbol{\mathcal{L}}}$.

Update $\widehat{\boldsymbol{\mathcal{S}}}$ with Projection. For updating $\widehat{\boldsymbol{\mathcal{S}}}$, the third optimization objective is:

$$\min_{\widehat{\boldsymbol{\mathcal{S}}}\in\mathcal{Q}} \frac{\lambda}{2}\|\boldsymbol{\mathcal{S}} - \widehat{\boldsymbol{\mathcal{S}}} + \boldsymbol{\mathcal{V}}\|_F^2. \tag{11}$$

Similar to the low-tensor-rank $\widehat{\boldsymbol{\mathcal{L}}}$, the sparse-constrained $\widehat{\boldsymbol{\mathcal{S}}}$ can also be analytically updated as

$$\widehat{\boldsymbol{\mathcal{S}}} \leftarrow \mathbf{proj}_{\mathcal{Q}}(\boldsymbol{\mathcal{S}} + \boldsymbol{\mathcal{V}}), \tag{12}$$

where $\mathbf{proj}(\cdot)$ is the projection that removes filters with the smallest ℓ_1 values to ensure that the updated $\widehat{\boldsymbol{\mathcal{S}}}$ can satisfy the structured sparse constraint.

Update Multipliers $\boldsymbol{\mathcal{U}}, \boldsymbol{\mathcal{V}}$. Upon updating $\widehat{\boldsymbol{\mathcal{L}}}$ and $\widehat{\boldsymbol{\mathcal{S}}}$, the dual multipliers $\boldsymbol{\mathcal{U}}$ and $\boldsymbol{\mathcal{V}}$ are updated as:

$$\boldsymbol{\mathcal{U}} \leftarrow \boldsymbol{\mathcal{U}} + \boldsymbol{\mathcal{L}} - \widehat{\boldsymbol{\mathcal{L}}}, \;\; \boldsymbol{\mathcal{V}} \leftarrow \boldsymbol{\mathcal{V}} + \boldsymbol{\mathcal{S}} - \widehat{\boldsymbol{\mathcal{S}}}. \tag{13}$$

Notice that after the iterative update finishes, the low-rank component $\boldsymbol{\mathcal{L}}$ is explicitly decomposed to TT-cores $\{\boldsymbol{\mathcal{C}}\}_{j=0}^{d}$, and the entire compressed model consisting of TT-cores and sparse part $\boldsymbol{\mathcal{S}}$ is finally fine-tuned with standard SGD. The overall CEPD algorithm is summarized in Algorithm 1.

5 EXPERIMENTS

Dataset and Baseline. We evaluate our proposed approach on two image classification datasets (CIFAR-10 and ImageNet). For experiments on CIFAR-10, four CNN models (ResNet-20, ResNet-56, DenseNet-40 and MobileNetV2) are compressed. For experiments on ImageNet, we evaluate our approach for ResNet-18/50 and compare it with state-of-the-art model compression methods.

Hyperparameter. All the experiments are conducted using SGD optimizer with batch size and momentum as 128, 0.9. The weight decay is set to 0.0001 for CIFAR-10 and 0.00002 for ImageNet respectively. The learning rates in the optimization and fine-tuning process are set as 0.1 and 0.01, respectively, and they gradually decrease following the cosine scheduler. The entire training procedure is performed on NVIDIA-V100 GPUs with PyTorch 1.12.

Results on CIFAR-10 Dataset. Table 1 shows the evaluation results on CIFAR-10. For each baseline model, we compare CEPD with several types of compression methods, including decomposition-only (***L***: PSTRN (Li et al. (2021a)), TRP (Xu et al. (2020)), SVDT (Yang et al. (2020)), LREL (Idelbayev & Carreira-Perpinán (2020)), ALDS (Liebenwein et al. (2021)), CaP (Minnehan & Savakis (2019)) and ENC-Inf (Kim et al. (2019))), pruning-only (***S***: SCOP (Tang et al. (2020)), FPGM (He et al. (2019)), HRank (Lin et al. (2020)), CHIP (Sui et al. (2021)), CHEX (Hou et al. (2022)), ISP (Ganjdanesh et al. (2022))), first-pruning-then-decomposition (***L(S)***: CC (Li et al. (2021b))), and layer-wise either-pruning-or-decomposition (***S/L***: Hinge (Li et al. (2020))). From this table it is seen that CEPD consistently outperforms structured pruning and low-rank decomposition works.

Results on ImageNet Dataset. Table 2 summarizes the compression performance of our approach and other existing works for ResNet-18/50 on ImageNet dataset. In addition to previously mentioned works, we compare our CEPD with three other $\boldsymbol{L}$ works, MetaP (Liu et al. (2019)), FR (Chu & Lee (2021)), and STABLE (Phan et al. (2020)), and some $\boldsymbol{S}$ works, FBS (Gao et al. (2019)), ISP (Ganjdanesh et al. (2022)). It is seen that our CEPD solution can bring 0.10% and 1.39% accuracy increase for ResNet-18 and ResNet-50 over baseline models with 59.4% and 54.6% fewer parameters, respectively. When targeting for generating more compact model of ResNet-50, our approach can still achieve high performance – it only has 0.31% accuracy drop with 63.3% model size reduction, which outperforms the other related works.

Algorithm 1 The overall CEPD algorithm

Input: Pre-trained weight tensor $\boldsymbol{\mathcal{W}}$, target TT-ranks $\{\gamma_j\}_{j=0}^{d}$, sparse target κ, training epochs T.
Output: TT-cores $\{\boldsymbol{\mathcal{C}}\}_{j=0}^{d}$, sparse component $\boldsymbol{\mathcal{S}}$.

Initialize $\boldsymbol{\mathcal{L}}, \widehat{\boldsymbol{\mathcal{L}}}, \boldsymbol{\mathcal{S}}, \widehat{\boldsymbol{\mathcal{S}}}$ with $\boldsymbol{\mathcal{W}}$;
Initialize $\boldsymbol{\mathcal{U}} := \boldsymbol{0}, \boldsymbol{\mathcal{V}} := \boldsymbol{0}$
for $t = 1$ **to** T **do**
  Update $\boldsymbol{\mathcal{U}}, \boldsymbol{\mathcal{V}}$ using Eq. 13;
  Update $\boldsymbol{\mathcal{L}}$ and $\boldsymbol{\mathcal{S}}$ using Eq. 5 and Eq. 6;
  // Update $\widehat{\boldsymbol{\mathcal{L}}}$ using TT-truncation
  $\widehat{\boldsymbol{\mathcal{L}}} \leftarrow \textbf{trunc}_{\mathcal{P}}(\boldsymbol{\mathcal{L}} + \boldsymbol{\mathcal{U}})$;
  // Update $\widehat{\boldsymbol{\mathcal{S}}}$ using projection
  $\widehat{\boldsymbol{\mathcal{S}}} \leftarrow \textbf{proj}_{\mathcal{Q}}(\boldsymbol{\mathcal{S}} + \boldsymbol{\mathcal{V}})$;
end for
Decompose $\boldsymbol{\mathcal{L}}$ to TT-cores $\{\boldsymbol{\mathcal{C}}\}_{j=0}^{d}$;
Fine-tune model with $\{\boldsymbol{\mathcal{C}}\}_{j=0}^{d}$ and $\boldsymbol{\mathcal{S}}$.

Table 1: Experiment results on CIFAR-10. "L" denotes low-rank decomposition; "S" denotes pruning. **No prior tensor decomposition work reports performance for ResNet-56 and DenseNet-40.**

Compression Method	Type	Decomp. Format	Top-1 Accuracy (%)			Params. ↓ (%)	FLOPs ↓ (%)
			Baseline	Comp.	Δ		
ResNet-20							
PSTRN	L	Tensor	91.25	90.80	-0.45	55.6	N/A
TRP	L	Matrix	91.74	90.88	-0.86	48.1	51.0
SVDT	L	Matrix	90.93	90.97	+0.04	N/A	54.5
LREL	L	Matrix	91.60	90.20	-1.40	N/A	66.7
ALDS	L	Matrix	91.39	90.92	-0.47	74.9	67.9
Hinge	S/L	Matrix	92.54	91.84	-0.70	55.5	54.5
SCOP	S	N/A	92.22	90.75	-1.47	56.3	55.7
FPGM	S	N/A	92.20	90.44	-1.76	51.0	54.0
CEPD (Ours)	L+S	Tensor	91.25	**91.91**	**+0.66**	**56.6**	**56.2**
CEPD (Ours)	L+S	Tensor	91.25	**91.02**	**-0.23**	**76.4**	**68.1**
ResNet-56							
TRP	L	Matrix	93.14	92.77	-0.37	N/A	56.7
CaP	L	Matrix	93.51	93.22	-0.29	N/A	49.8
ENC-Inf	L	Matrix	93.10	93.00	-0.10	N/A	50.0
HRank	S	N/A	93.26	93.52	+0.26	16.8	29.3
HRank	S	N/A	93.26	93.17	-0.09	42.4	50.0
CC	L(S)	Matrix	93.33	93.87	+0.54	36.5	42.4
CC	L(S)	Matrix	93.33	93.64	+0.31	48.2	52.0
CEPD (Ours)	L+S	Tensor	93.27	**93.99**	**+0.72**	**41.5**	**43.0**
CEPD (Ours)	L+S	Tensor	93.27	**93.70**	**+0.43**	**63.6**	**53.1**
DenseNet-40							
HRank	S	N/A	94.81	94.24	-0.57	36.5	40.8
HRank	S	N/A	94.81	93.68	-1.13	53.8	61.0
Hinge	S/L	Matrix	94.74	94.67	-0.07	27.5	44.4
CC	L(S)	Matrix	94.81	94.67	-0.14	51.9	47.0
CC	L(S)	Matrix	94.81	94.40	-0.41	64.4	60.4
CEPD (Ours)	L+S	Tensor	94.81	**94.79**	**-0.02**	**52.6**	**50.3**
CEPD (Ours)	L+S	Tensor	94.81	**94.55**	**-0.26**	**65.3**	**62.1**
MobilenetV2							
Uniform	S	N/A	94.47	94.17	-0.30	23.6	26.4
DCP	S	N/A	94.47	94.69	+0.22	23.6	26.4
SCOP	S	N/A	94.48	94.24	-0.24	36.1	40.3
ISP	S	N/A	94.53	94.85	+0.32	N/A	44.0
CEPD (Ours)	L+S	Tensor	94.48	**94.93**	**+0.45**	**48.6**	**44.2**

Table 2: Experiment results on ImageNet. "L" denotes low-rank decomposition; "S" denotes pruning.

Compression Method	Type	Decomp. Format	Top-1 Accuracy (%)			Top-5 Accuracy (%)			Params. ↓(%)	FLOPs ↓(%)
			Base.	Comp.	Δ	Base.	Comp.	Δ		
ResNet-18										
TRP	L	Matrix	69.10	65.46	-3.64	88.94	86.48	-2.46	N/A	44.8
ALDS	L	Matrix	69.62	69.70	+0.08	89.08	89.26	+0.18	66.7	43.5
FR	L	Tensor	69.76	69.04	-0.72	N/A	N/A	N/A	57.9	50.5
SCOP	S	N/A	69.76	68.62	-1.14	89.08	88.45	-0.63	43.5	45.0
CEPD (Ours)	L+S	Tensor	69.76	**69.86**	**+0.10**	89.08	**89.32**	**+0.24**	**59.4**	**51.3**
ResNet-50										
TRP	L	Matrix	75.90	74.06	-1.84	92.70	92.07	-0.63	N/A	44.4
SVDT	L	Matrix	N/A	N/A	N/A	91.91	91.97	+0.06	N/A	30.6
MetaP	S	N/A	76.60	75.40	-1.20	N/A	N/A	N/A	N/A	51.6
SCOP	S	N/A	76.15	75.95	-0.20	92.87	92.79	-0.08	42.8	45.3
CHIP	S	N/A	76.15	76.30	+0.15	92.87	93.02	+0.15	40.8	44.8
ISP	S	N/A	76.13	75.97	-0.16	92.86	92.74	-0.12	N/A	56.6
CHEX	S	N/A	N/A	77.40	N/A	N/A	N/A	N/A	N/A	51.6
CC	L(S)	Matrix	76.15	75.59	-0.56	92.87	92.64	-0.23	48.4	52.9
CEPD (Ours)	L+S	Tensor	76.13	**77.52**	**+1.39**	92.86	**94.00**	**+1.24**	**54.6**	**53.9**
TRP	L	Matrix	75.90	72.69	-3.21	92.70	91.41	-1.29	N/A	56.5
STABLE	L	Tensor	76.15	74.68	-1.47	92.87	92.16	-0.71	60.2	62.1
HRank	S	N/A	76.15	71.98	-4.17	92.87	91.01	-1.86	46.0	62.1
CHIP	S	N/A	76.15	75.26	-0.89	92.87	92.53	-0.34	56.7	62.8
Hinge	S/L	Matrix	76.15	74.70	-1.45	N/A	N/A	N/A	N/A	53.5
CC	L(S)	Matrix	76.15	74.54	-1.61	92.87	92.25	-0.62	58.6	62.7
CEPD (Ours)	L+S	Tensor	76.13	**75.82**	**-0.31**	92.86	**92.84**	**-0.02**	**63.3**	**62.9**

Comparison with Other Additive Compression Methods. Because existing ***L+S*** works ((LRSD (Guo et al. (2019)) and UAF (Ma et al. (2019))) are not evaluated in the modern DNN models on large-scale datasets (e.g., ResNet on ImageNet), we compare them with CEPD in a separate Table 3. It is seen that CEPD achieves better performance with the same or even higher compression ratio.

Table 3: Comparison with existing additive compression methods on CIFAR-10.

Method	Model	Type	Acc. (%)	Params. ↓
UAF	VGG -16	L+S	91.65	77.48%
Ours			**92.33**	**78.26%**
LRSD	VGGNet -7	L+S	86.17	76.09%
Ours			**86.30**	**76.13%**

Inference Speedup. We evaluate the inference speedup of CEPD for compressing ResNet-50 on FPGA and ASIC platforms, targeting to different settings of FLOPs reduction. As illustrated in Table 4, the compressed models exhibiting both sparsity and low-rankness enjoy practical speedup.

Table 4: Inference time (per image) for the compressed ResNet-50 via using CEPD.

	FLOPs Remaining	FPGA Xilinx PYNQZ1	ASIC Eyeriss
Baseline	100%	172.0ms (1×)	38.10ms (1×)
CEPD (Ours)	66.3%	**122.0ms (1.41×)**	**28.26ms (1.35×)**
CEPD (Ours)	46.1%	**88.21ms (1.95×)**	**20.88ms (1.82×)**
CEPD (Ours)	37.1%	**67.72ms (2.54×)**	**15.83ms (2.41×)**

6 CONCLUSION

In this paper we propose to systematically co-explore DNN low-rankness and sparsity for efficient model compression. By performing comprehensive analysis on critical design factors, we propose CEPD, a unified compression framework that can capture model low-rankness and sparsity simultaneously and efficiently. Evaluation results show that our proposed approach can bring significant model size and computational cost reductions while still preserving high model accuracy.

REFERENCES

Thierry Bouwmans, Necdet Serhat Aybat, and El-hadi Zahzah. *Handbook of robust low-rank and sparse matrix decomposition: Applications in image and video processing*. CRC Press, 2016.

Bo-Shiuan Chu and Che-Rung Lee. Low-rank tensor decomposition for compression of convolutional neural networks using funnel regularization. *arXiv preprint arXiv:2112.03690*, 2021.

Emily L Denton, Wojciech Zaremba, Joan Bruna, Yann LeCun, and Rob Fergus. Exploiting linear structure within convolutional networks for efficient evaluation. In *Advances in neural information processing systems*, pp. 1269–1277, 2014.

Abhimanyu Dubey, Moitreya Chatterjee, and Narendra Ahuja. Coreset-based neural network compression. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 454–470, 2018.

Alireza Ganjdanesh, Shangqian Gao, and Heng Huang. Interpretations steered network pruning via amortized inferred saliency maps. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2022.

Shangqian Gao, Feihu Huang, Weidong Cai, and Heng Huang. Network pruning via performance maximization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 9270–9280, 2021.

Xitong Gao, Yiren Zhao, Łukasz Dudziak, Robert Mullins, and Cheng zhong Xu. Dynamic channel pruning: Feature boosting and suppression. In *International Conference on Learning Representations*, 2019.

Kailing Guo, Xiaona Xie, Xiangmin Xu, and Xiaofen Xing. Compressing by learning in a low-rank and sparse decomposition form. *IEEE Access*, 7:150823–150832, 2019.

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.

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

Zejiang Hou, Minghai Qin, Fei Sun, Xiaolong Ma, Kun Yuan, Yi Xu, Yen-Kuang Chen, Rong Jin, Yuan Xie, and Sun-Yuan Kung. Chex: Channel exploration for cnn model compression. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 12287–12298, 2022.

Yerlan Idelbayev and Miguel A Carreira-Perpinán. Low-rank compression of neural nets: Learning the rank of each layer. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 8049–8059, 2020.

Yerlan Idelbayev and Miguel A Carreira-Perpinán. More general and effective model compression via an additive combination of compressions. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pp. 233–248. Springer, 2021.

Hyeji Kim, Muhammad Umar Karim Khan, and Chong-Min Kyung. Efficient neural network compression. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 12569–12577, 2019.

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. In *International Conference on Learning Representations*, 2016.

Jean Kossaifi, Adrian Bulat, Georgios Tzimiropoulos, and Maja Pantic. T-net: Parametrizing fully convolutional nets with a single high-order tensor. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 7822–7831, 2019.

Jean Kossaifi, Antoine Toisoul, Adrian Bulat, Yannis Panagakis, Timothy M Hospedales, and Maja Pantic. Factorized higher-order cnns with an application to spatio-temporal emotion estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 6060–6069, 2020.

Chong Li and CJ Shi. Constrained optimization based low-rank approximation of deep neural networks. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 732–747, 2018.

Nannan Li, Yu Pan, Yaran Chen, Zixiang Ding, Dongbin Zhao, and Zenglin Xu. Heuristic rank selection with progressively searching tensor ring network. *Complex & Intelligent Systems*, pp. 1–15, 2021a.

Yawei Li, Shuhang Gu, Christoph Mayer, Luc Van Gool, and Radu Timofte. Group sparsity: The hinge between filter pruning and decomposition for network compression. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 8018–8027, 2020.

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

Lucas Liebenwein, Cenk Baykal, Harry Lang, Dan Feldman, and Daniela Rus. Provable filter pruning for efficient neural networks. *arXiv preprint arXiv:1911.07412*, 2019.

Lucas Liebenwein, Alaa Maalouf, Dan Feldman, and Daniela Rus. Compressing neural networks: Towards determining the optimal layer-wise decomposition. *Advances in Neural Information Processing Systems*, 34, 2021.

Mingbao Lin, Rongrong Ji, Yan Wang, Yichen Zhang, Baochang Zhang, Yonghong Tian, and Ling Shao. Hrank: Filter pruning using high-rank feature map. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 1529–1538, 2020.

Zechun Liu, Haoyuan Mu, Xiangyu Zhang, Zichao Guo, Xin Yang, Kwang-Ting Cheng, and Jian Sun. Metapruning: Meta learning for automatic neural network channel pruning. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 3296–3305, 2019.

Xi Luo. Recovering model structures from large low rank and sparse covariance matrix estimation. *arXiv preprint arXiv:1111.1133*, 2011.

Yuzhe Ma, Ran Chen, Wei Li, Fanhua Shang, Wenjian Yu, Minsik Cho, and Bei Yu. A unified approximation framework for compressing and accelerating deep neural networks. In *2019 IEEE 31st International Conference on Tools with Artificial Intelligence (ICTAI)*, pp. 376–383. IEEE, 2019.

Breton Minnehan and Andreas Savakis. Cascaded projection: End-to-end network compression and acceleration. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 10715–10724, 2019.

Alexander Novikov, Dmitrii Podoprikhin, Anton Osokin, and Dmitry P Vetrov. Tensorizing neural networks. *Advances in Neural Information Processing Systems*, 28:442–450, 2015.

Anh-Huy Phan, Konstantin Sobolev, Konstantin Sozykin, Dmitry Ermilov, Julia Gusak, Petr Tichavskỳ, Valeriy Glukhov, Ivan Oseledets, and Andrzej Cichocki. Stable low-rank tensor decomposition for compression of convolutional neural network. In *European Conference on Computer Vision*, pp. 522–539. Springer, 2020.

Emile Richard, Pierre-André Savalle, and Nicolas Vayatis. Estimation of simultaneously sparse and low rank matrices. *arXiv preprint arXiv:1206.6474*, 2012.

Emile Richard, BACH Francis, and Jean-Philippe Vert. Intersecting singularities for multi-structured estimation. In *International Conference on Machine Learning*, pp. 1157–1165. PMLR, 2013.

Yang Sui, Miao Yin, Yi Xie, Huy Phan, Saman Aliari Zonouz, and Bo Yuan. Chip: Channel independence-based pruning for compact neural networks. *Advances in Neural Information Processing Systems*, 34, 2021.

Cheng Tai, Tong Xiao, Yi Zhang, Xiaogang Wang, et al. Convolutional neural networks with low-rank regularization. In *International Conference on Learning Representations*, 2016.

Yehui Tang, Yunhe Wang, Yixing Xu, Dacheng Tao, Chunjing XU, Chao Xu, and Chang Xu. Scop: Scientific control for reliable neural network pruning. In *Advances in Neural Information Processing Systems*, volume 33, pp. 10936–10947, 2020.

Wenqi Wang, Yifan Sun, Brian Eriksson, Wenlin Wang, and Vaneet Aggarwal. Wide compression: Tensor ring nets. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 9329–9338, 2018.

Zi Wang, Chengcheng Li, and Xiangyang Wang. Convolutional neural network pruning with structural redundancy reduction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 14913–14922, 2021.

Yuhui Xu, Yuxi Li, Shuai Zhang, Wei Wen, Botao Wang, Yingyong Qi, Yiran Chen, Weiyao Lin, and Hongkai Xiong. Trp: Trained rank pruning for efficient deep neural networks. In *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI-20*, pp. 977–983, 2020.

Huanrui Yang, Minxue Tang, Wei Wen, Feng Yan, Daniel Hu, Ang Li, Hai Li, and Yiran Chen. Learning low-rank deep neural networks via singular vector orthogonality regularization and singular value sparsification. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition workshops*, pp. 678–679, 2020.

Xiyu Yu, Tongliang Liu, Xinchao Wang, and Dacheng Tao. On compressing deep models by low rank and sparse decomposition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 7370–7379, 2017.

APPENDIX

A DIFFERENCE BETWEEN 2-D MATRIX DECOMPOSITION AND 4-D TENSOR DECOMPOSITION

As illustrated in Figure 4, for an example convolutional layer, imposing the low-rank structure can be realized by performing simple 2-D matrix factorization or advanced high-order tensor decomposition.

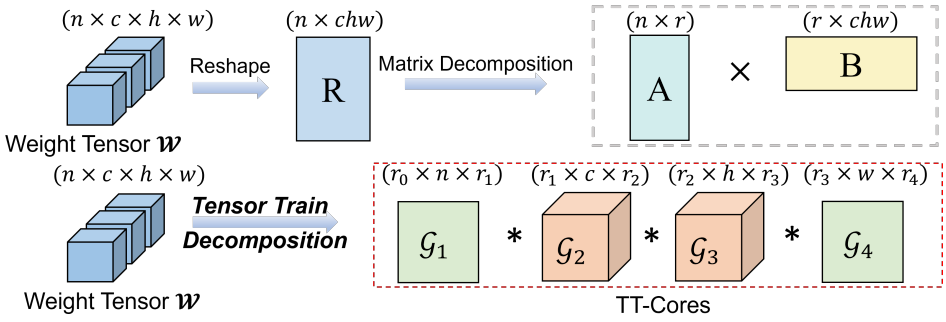

Figure 4: Exploring low-rankness of convolutional layer via matrix decomposition (Top) and tensor decomposition (Bottom). Here tensor train (TT) decomposition is adopted for illustration.

B EFFECT OF CO-EXPLORATION PROCEDURE

Simultaneously Obtaining Low-rankness and Sparsity. Figure 5 shows the loss curves during the model optimization procedure. Notice that here besides overall training loss, the individual low-rank and sparse loss component, which directly reflects the progress of enforcing low-rankness and sparsity, respectively, is also explicitly reported in this figure. It is seen that our proposed approach indeed successfully and simultaneously imposes the desired low-tensor-rankness and sparsity with hard constraints onto the model, and thereby ensuring that the compressed model can fully exhibit both low-rank and sparse characteristics after the optimization.

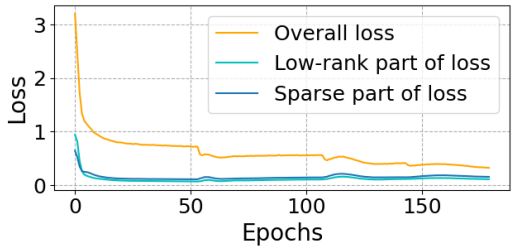

Figure 5: Loss curves of a ResNet-20 trained on CIFAR-10 dataset using our CEPD algorithm. Here the curves of the individual low-rank and sparse loss components are also illustrated. **It is seen that the low-rankness and sparsity are indeed imposed on the model via using CEPD.**

Effect of Optimization Procedure. We also study the benefits of using our proposed optimization procedure described in Algorithm 1 to solve problem (2). Here we compare our approach with a ***direct method*** that performs TT decomposition and pruning on the uncompressed model straightforwardly with the same TT-ranks and sparsity settings. In addition, the same fine-tuning process that CEPD adopts is also applied in this direct method. Figure 6 shows the comparison results with respect to different compression ratios. It is seen that our proposed optimization procedure brings significant accuracy increase.

Visualization. Figure 7 visualizes the weight tensor of one convolutional layer in a pre-trained ResNet-20 model before and after performing our proposed compression approach. Here the visualization of

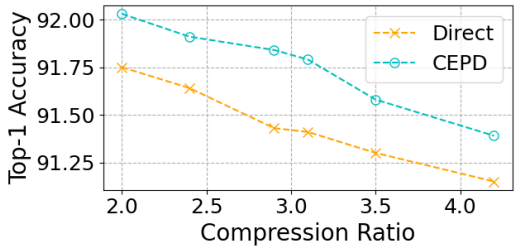

Figure 6: The effect of optimization procedure for jointly TT decomposing and pruning ResNet-20 on CIFAR10. Here CEPD and the direct method use the same TT-rank setting and sparsity ratio. However, the direct method does not perform optimization on original model. Instead, it first performs TT decomposition and then prunes the difference between original model and the low-rank component to obtain the sparse component. The two components generated by this direct method will then be fine-tuned with the same way that CEPD uses. **It is seen that our proposed optimization procedure in CEPD brings significantly accuracy increase.**

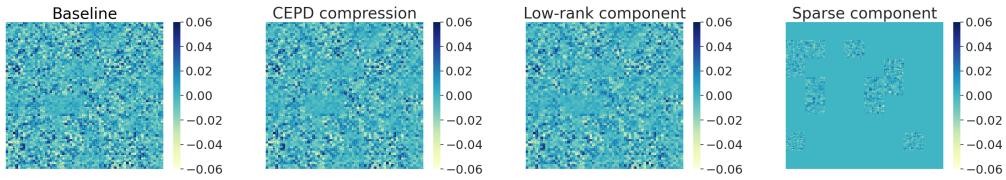

Figure 7: Visualization of one layer of ResNet-20 before and after performing our proposed CEPD compression. Here the low-rank and sparse components of the compressed layer are also visualized. **It is seen that low-rank component preserve most of weight information, and some spatial patterns are contained in the sparse component.**

the low-rank and sparse components of the compressed layer is also illustrated in this figure. It is seen that most of the weight information is preserved in the low-rank component, and meanwhile the sparse component contains some spatial pattern as well.

C Empirical Analysis & Ablation Study

Selection of Rank Value/Sparsity Ratio. Conventionally the rank/sparsity selection is costly since combinatorial search for all layers is required. One benefit of our proposed approach is its low-cost determination process. This is because many or even all of the layers **share the same rank value and sparsity ratio.** We report the detailed layer-wise rank and sparsity values for the models we evaluate in **Appendix** E. Here the ranks of the layers in the same residual block can be simply set as same, and the sparsity of all layers can be set as same.

Compression Ratio-vs-Model Accuracy. We study the accuracy performance of compressed model (ResNet-20 on CIFAR-10) with different compression ratios. Here two different cases, keeping sparsity with changing rank and keeping rank with changing sparsity, are studied and reported in Table 13 and 14 (see **Appendix** F), respectively.

More Low-Rankness or More Sparsity? We also empirically study the effect of targeting higher low-rankness or higher sparsity with the same overall compression ratio. Experimental results are reported in Table 15 in **Appendix** F. It is seen that the different configurations of low-rankness and sparsity have very similar effects on the overall accuracy, unless the sparsity is extremely high.

D DETAILS OF ANALYSIS IN SECTION 3

D.1 MORE RESULTS FOR ANALYSIS IN QUESTION #1

Table 5,6 and 7 show the layer-wise approximation errors incurred by three operational sequences (***L+S***, ***S(L)*** and ***L(S)***). Here the baseline models include well-trained ResNet-20, ResNet-56 on CIFAR-10, and ResNet-50 on ImageNet, and the compression ratio is set as 3.0 for all the layers. **It is seen that *L+S* scheme always brings the smallest approximation errors.**

Table 5: Approximation errors for different layers of ResNet-20 with different operational sequences. The compression ratio is set as 3.0 for all the layers.

Layer Name	Weight Shape	Approximation Errors		
		L+S	S(L)	L(S)
l1b1.conv1	(16, 16, 3, 3)	1.95	2.17	1.96
l1b1.conv2	(16, 16, 3, 3)	1.83	2.04	1.84
l1b2.conv1	(16, 16, 3, 3)	1.78	2.02	1.83
l1b2.conv2	(16, 16, 3, 3)	1.67	1.89	1.69
l1b3.conv1	(16, 16, 3, 3)	2.25	2.58	2.35
l1b3.conv2	(16, 16, 3, 3)	1.81	1.85	1.81
l2b1.conv1	(16, 32, 3, 3)	2.44	2.74	2.52
l2b1.conv2	(32, 32, 3, 3)	3.04	3.34	3.04
l2b2.conv1	(32, 32, 3, 3)	2.99	3.39	3.04
l2b2.conv2	(32, 32, 3, 3)	2.70	2.95	2.71
l2b3.conv1	(32, 32, 3, 3)	2.96	3.35	2.99
l2b3.conv2	(32, 32, 3, 3)	2.49	2.78	2.50
l3b1.conv1	(32, 64, 3, 3)	3.74	4.21	3.78
l3b1.conv2	(64, 64, 3, 3)	4.92	5.26	4.92
l3b2.conv1	(64, 64, 3, 3)	5.17	5.63	5.23
l3b2.conv2	(64, 64, 3, 3)	4.27	4.76	4.27
l3b3.conv1	(64, 64, 3, 3)	4.79	5.21	4.80
l3b3.conv2	(64, 64, 3, 3)	0.48	1.97	1.00

D.2 MORE RESULTS FOR ANALYSIS IN QUESTION #2

Table 8 shows the difference (in term of mean square error (MSE)) between the output feature maps of original layer and the compressed one in ResNet-20 on CIFAR-10 dataset. Here SVD and tensor train (TT) are adopted for matrix decomposition and tensor decomposition, respectively. **It is seen that, with the same or even higher compression ratio, high-order tensor decomposition always brings smaller approximation error than the matrix decomposition..**

Table 9 shows the difference between the output feature maps of one original layer and the compressed version in ResNet-20 on CIFAR-10 dataset and the difference of the final accuracy. Here SVD, Tucker and TT are adopted for low-rank decompositions. **It is seen that high-order tensor decomposition always brings smaller approximation error than the matrix decomposition and TT always has the least impact on the final accuracy.**

Table 6: Approximation errors for different layers of ResNet-56 with different operational sequences. The compression ratio is set as 3.0 for all the layers.

Layer Name	Weight Shape	Approximation Errors		
		L+S	S(L)	L(S)
l1b1.conv1	(16, 16, 3, 3)	2.51	3.05	2.51
l1b1.conv2	(16, 16, 3, 3)	2.72	3.15	2.79
l1b2.conv1	(16, 16, 3, 3)	3.18	3.64	3.23
l1b2.conv2	(16, 16, 3, 3)	2.99	3.47	3.04
l1b3.conv1	(16, 16, 3, 3)	2.70	3.17	2.75
l1b3.conv2	(16, 16, 3, 3)	2.80	3.20	2.86
l1b4.conv1	(16, 16, 3, 3)	2.71	3.11	2.76
l1b4.conv2	(16, 16, 3, 3)	2.70	2.86	2.78
l1b5.conv1	(16, 16, 3, 3)	3.04	3.32	3.05
l1b5.conv2	(16, 16, 3, 3)	2.47	2.81	2.52
l1b6.conv1	(16, 16, 3, 3)	2.54	2.97	2.55
l1b6.conv2	(16, 16, 3, 3)	2.32	2.71	2.35
l1b7.conv1	(16, 16, 3, 3)	3.53	4.14	3.70
l1b7.conv2	(16, 16, 3, 3)	2.63	2.83	2.64
l1b8.conv1	(16, 16, 3, 3)	3.37	3.92	3.37
l1b8.conv2	(16, 16, 3, 3)	2.50	2.84	2.53
l1b9.conv1	(16, 16, 3, 3)	2.45	2.90	2.46
l1b9.conv2	(16, 16, 3, 3)	1.94	2.50	1.97
l2b1.conv1	(32, 16, 3, 3)	4.49	4.87	4.60
l2b1.conv2	(32, 32, 3, 3)	5.22	6.02	5.23
l2b2.conv1	(32, 32, 3, 3)	3.65	4.47	3.70
l2b2.conv2	(32, 32, 3, 3)	3.57	4.23	3.58
l2b3.conv1	(32, 32, 3, 3)	4.36	5.06	4.37
l2b3.conv2	(32, 32, 3, 3)	4.16	4.62	4.17
l2b4.conv1	(32, 32, 3, 3)	4.31	4.86	4.32
l2b4.conv2	(32, 32, 3, 3)	3.91	4.53	3.96
l2b5.conv1	(32, 32, 3, 3)	4.27	5.24	4.27
l2b5.conv2	(32, 32, 3, 3)	3.73	4.22	3.73
l2b6.conv1	(32, 32, 3, 3)	4.30	4.85	4.30
l2b6.conv2	(32, 32, 3, 3)	3.79	4.25	3.79
l2b7.conv1	(32, 32, 3, 3)	4.24	4.56	4.26
l2b7.conv2	(32, 32, 3, 3)	3.52	3.89	3.55
l2b8.conv1	(32, 32, 3, 3)	4.20	4.66	4.20
l2b8.conv2	(32, 32, 3, 3)	3.59	4.02	3.65
l2b9.conv1	(32, 32, 3, 3)	4.30	4.69	4.30
l2b9.conv2	(32, 32, 3, 3)	3.22	3.76	3.26
l3b1.conv1	(64, 32, 3, 3)	6.51	7.36	6.53
l3b1.conv2	(64, 64, 3, 3)	8.09	8.67	8.13
l3b2.conv1	(64, 64, 3, 3)	7.17	8.12	7.25
l3b2.conv2	(64, 64, 3, 3)	7.30	7.91	7.31
l3b3.conv1	(64, 64, 3, 3)	7.20	7.95	7.23
l3b3.conv2	(64, 64, 3, 3)	6.72	7.45	6.75
l3b4.conv1	(64, 64, 3, 3)	8.07	8.58	8.08
l3b4.conv2	(64, 64, 3, 3)	6.72	7.41	6.72
l3b5.conv1	(64, 64, 3, 3)	8.07	8.70	8.08
l3b5.conv2	(64, 64, 3, 3)	6.14	6.88	6.15
l3b6.conv1	(64, 64, 3, 3)	7.81	8.41	7.82
l3b6.conv2	(64, 64, 3, 3)	5.18	5.87	5.25
l3b7.conv1	(64, 64, 3, 3)	6.54	7.12	6.54
l3b7.conv2	(64, 64, 3, 3)	4.08	4.72	4.09
l3b8.conv1	(64, 64, 3, 3)	5.02	5.66	5.03
l3b8.conv2	(64, 64, 3, 3)	2.64	3.19	2.64
l3b9.conv1	(64, 64, 3, 3)	4.05	5.05	4.06
l3b9.conv2	(64, 64, 3, 3)	1.73	2.75	1.80

Table 7: Approximation errors for different layers of ResNet-50 with different operational sequences. The compression ratio is set as 3.0 for all the layers.

Layer Name	Weight Shape	Approximation Errors		
		L+S	S(L)	L(S)
l1b1.conv1	(64, 64, 1, 1)	2.62	2.74	2.65
l1b1.conv2	(64, 64, 3, 3)	2.10	4.02	2.12
l1b1.conv3	(256, 64, 1, 1)	1.87	3.20	1.99
l1b2.conv1	(64, 256, 1, 1)	2.00	2.78	2.03
l1b2.conv2	(64, 64, 3, 3)	3.03	3.91	3.04
l1b2.conv3	(256, 64, 1, 1)	2.43	2.84	2.44
l1b3.conv1	(64, 256, 1, 1)	1.97	2.66	2.06
l1b3.conv2	(64, 64, 3, 3)	3.62	4.57	3.77
l1b3.conv3	(256, 64, 1, 1)	2.31	2.65	2.31
l2b1.conv1	(128, 256, 1, 1)	3.37	4.53	3.58
l2b1.conv2	(128, 128, 3, 3)	5.32	6.43	5.49
l2b1.conv3	(512, 128, 1, 1)	4.01	4.53	4.07
l2b2.conv1	(128, 512, 1, 1)	1.35	3.50	1.74
l2b2.conv2	(128, 128, 3, 3)	2.34	5.69	2.55
l2b2.conv3	(512, 128, 1, 1)	2.22	3.65	2.24
l2b3.conv1	(128, 512, 1, 1)	3.17	4.51	3.35
l2b3.conv2	(128, 128, 3, 3)	4.63	6.25	4.85
l2b3.conv3	(512, 128, 1, 1)	3.99	4.70	4.00
l2b4.conv1	(128, 512, 1, 1)	3.53	4.50	3.77
l2b4.conv2	(128, 128, 3, 3)	5.44	6.49	5.54
l2b4.conv3	(512, 128, 1, 1)	3.82	4.23	3.79
l3b1.conv1	(256, 512, 1, 1)	6.38	8.20	6.83
l3b1.conv2	(256, 256, 3, 3)	7.66	9.51	7.93
l3b1.conv3	(1024, 256, 1, 1)	7.07	8.03	7.09
l3b2.conv1	(256, 1024, 1, 1)	3.98	5.56	4.35
l3b2.conv2	(256, 256, 3, 3)	6.23	8.60	6.58
l3b2.conv3	(1024, 256, 1, 1)	6.23	7.82	6.38
l3b3.conv1	(256, 1024, 1, 1)	4.17	6.11	4.66
l3b3.conv2	(256, 256, 3, 3)	6.59	8.70	6.95
l3b3.conv3	(1024, 256, 1, 1)	5.71	6.96	5.88
l3b4.conv1	(256, 1024, 1, 1)	5.29	6.80	5.63
l3b4.conv2	(256, 256, 3, 3)	7.13	8.79	7.34
l3b4.conv3	(1024, 256, 1, 1)	5.74	6.66	5.80
l3b5.conv1	(256, 1024, 1, 1)	5.83	7.16	6.14
l3b5.conv2	(256, 256, 3, 3)	7.14	8.99	7.37
l3b5.conv3	(1024, 256, 1, 1)	5.69	6.87	5.76
l3b6.conv1	(256, 1024, 1, 1)	6.65	7.81	6.89
l3b6.conv2	(256, 256, 3, 3)	7.23	9.18	7.57
l3b6.conv3	(1024, 256, 1, 1)	6.07	7.01	6.20
l4b1.conv1	(512, 1024, 1, 1)	10.99	13.57	11.68
l4b1.conv2	(512, 512, 3, 3)	11.37	14.41	12.19
l4b1.conv3	(2048, 512, 1, 1)	10.11	11.27	10.60
l4b2.conv1	(512, 2048, 1, 1)	9.65	11.21	10.15
l4b2.conv2	(512, 512, 3, 3)	12.10	14.46	12.67
l4b2.conv3	(2048, 512, 1, 1)	9.71	11.20	10.23
l4b3.conv1	(512, 2048, 1, 1)	12.51	14.14	12.86
l4b3.conv2	(512, 512, 3, 3)	10.30	12.44	10.83
l4b3.conv3	(2048, 512, 1, 1)	8.81	9.93	9.21

Table 8: Approximation errors for the feature maps of different layers of ResNet-20 on CIFAR-10 dataset with matrix decomposition (SVD) and tensor decomposition (TT).

Layer Name	Weight Shape	Compression Ratio		Approximation Error (MSE)	
		SVD	TT	SVD	TT
l1b1.conv1	(16, 16, 3, 3)	1.59	1.59	14.16	11.96
l1b1.conv2	(16, 16, 3, 3)	1.59	1.59	8.37	7.91
l1b2.conv1	(16, 16, 3, 3)	1.79	1.85	25.35	13.79
l1b2.conv2	(16, 16, 3, 3)	1.79	1.85	8.49	5.72
l1b3.conv1	(16, 16, 3, 3)	1.79	1.85	27.20	26.61
l1b3.conv2	(16, 16, 3, 3)	1.79	1.85	8.22	6.57
l2b1.conv1	(16, 32, 3, 3)	2.00	2.11	15.40	10.13
l2b1.conv2	(32, 32, 3, 3)	1.91	2.03	8.81	5.17
l2b2.conv1	(32, 32, 3, 3)	1.91	2.03	15.12	12.72
l2b2.conv2	(32, 32, 3, 3)	1.91	2.03	4.13	2.99
l2b3.conv1	(32, 32, 3, 3)	1.91	2.03	14.88	11.97
l2b3.conv2	(32, 32, 3, 3)	1.91	2.03	3.23	2.18
l3b1.conv1	(32, 64, 3, 3)	2.01	2.01	10.52	2.95
l3b1.conv2	(64, 64, 3, 3)	1.98	2.01	5.14	3.85
l3b2.conv1	(64, 64, 3, 3)	1.98	2.01	12.42	8.52
l3b2.conv2	(64, 64, 3, 3)	1.98	2.01	2.70	2.04
l3b3.conv1	(64, 64, 3, 3)	1.98	2.01	12.46	6.72
l3b3.conv2	(64, 64, 3, 3)	1.98	2.01	0.21	0.16

Table 9: Feature map approximation error of layer3.0.conv2 in ResNet-20 and the corresponding accuracy drop with SVD, Tucker and TT in different compression ratio settings.

SVD			Tucker			TT		
Compr. Ratio	Approx. Error	Acc. Δ (%)	Compr. Ratio	Approx. Error	Acc. Δ (%)	Compr. Ratio	Approx. Error	Acc. Δ (%)
1.47×	4.29	-1.06	1.48×	3.72	-0.43	1.51×	3.16	-0.31
1.98×	5.60	-3.0	1.97×	4.80	-2.1	2.01×	4.16	-0.8
2.50×	6.57	-5.27	2.50×	5.41	-3.46	2.50×	5.13	-1.86
3.03×	7.35	-7.62	3.09×	6.02	-3.76	3.09×	6.06	-3.18
3.38×	7.71	-9.66	3.46×	6.42	-4.54	3.57×	6.40	-4.03
4.11×	8.31	-11.32	4.18×	6.81	-6.04	4.20×	6.94	-5.41

E DISTRIBUTION OF TT-RANK AND SPARSITY RATIO

Table. 10, 11, and 12 list the layer-wise TT-rank and sparsity ratio of compressed models. Here Model I and Model II in each table correspond to two reported models of each network with different Top-1 accuracy in Table 1 and 2.

Table 10: Layer-wise distribution of TT-Rank and sparsity ratio for the compressed ResNet-20 models on CIFAR-10 dataset.

Layer Name	Weight Shape	Model I with 91.02% Acc.		Model II with 91.91% Acc.	
		TT Ranks	Sparsity (%)	TT Ranks	Sparsity (%)
l1b1.conv1	(16, 16, 3, 3)	(1, 8, 8, 1)	90.0	(1, 10, 10, 1)	90.0
l1b1.conv2	(16, 16, 3, 3)	(1, 8, 8, 1)	90.0	(1, 10, 10, 1)	90.0
l1b2.conv1	(16, 16, 3, 3)	(1, 8, 8, 1)	90.0	(1, 10, 10, 1)	90.0
l1b2.conv2	(16, 16, 3, 3)	(1, 8, 8, 1)	90.0	(1, 8, 8, 1)	90.0
l1b3.conv1	(16, 16, 3, 3)	(1, 8, 8, 1)	90.0	(1, 8, 8, 1)	90.0
l1b3.conv2	(16, 16, 3, 3)	(1, 10, 10, 1)	90.0	(1, 8, 8, 1)	90.0
l2b1.conv1	(16, 32, 3, 3)	(1, 10, 10, 1)	90.0	(1, 12, 12, 1)	90.0
l2b1.conv2	(32, 32, 3, 3)	(1, 10, 10, 1)	90.0	(1, 12, 12, 1)	90.0
l2b2.conv1	(32, 32, 3, 3)	(1, 10, 10, 1)	90.0	(1, 12, 12, 1)	90.0
l2b2.conv2	(32, 32, 3, 3)	(1, 10, 10, 1)	90.0	(1, 12, 12, 1)	90.0
l2b3.conv1	(32, 32, 3, 3)	(1, 10, 10, 1)	90.0	(1, 12, 12, 1)	90.0
l2b3.conv2	(32, 32, 3, 3)	(1, 16, 16, 1)	90.0	(1, 20, 20, 1)	90.0
l3b1.conv1	(32, 64, 3, 3)	(1, 16, 16, 1)	90.0	(1, 20, 20, 1)	90.0
l3b1.conv2	(64, 64, 3, 3)	(1, 16, 16, 1)	90.0	(1, 20, 20, 1)	90.0
l3b2.conv1	(64, 64, 3, 3)	(1, 16, 16, 1)	90.0	(1, 20, 20, 1)	90.0
l3b2.conv2	(64, 64, 3, 3)	(1, 16, 16, 1)	90.0	(1, 20, 20, 1)	90.0
l3b3.conv1	(64, 64, 3, 3)	(1, 16, 16, 1)	90.0	(1, 20, 20, 1)	90.0
l3b3.conv2	(64, 64, 3, 3)	(1, 16, 16, 1)	90.0	(1, 20, 20, 1)	90.0

Table 11: Layer-wise distribution of TT-Rank and sparsity ratio for the compressed DenseNet-40 models on CIFAR-10 dataset.

Layer Name	Weight Shape	Model I with 94.79% Acc.		Model II with 94.55% Acc.	
		TT Ranks	Sparsity(%)	TT Ranks	Sparsity(%)
b1l0.conv1	(12, 24, 3, 3)	(1, 18, 16, 1)	90.0	(1, 18, 16, 1)	90.0
b1l1.conv1	(12, 36, 3, 3)	(1, 20, 18, 1)	90.0	(1, 20, 18, 1)	90.0
b1l2.conv1	(12, 48, 3, 3)	(1, 30, 18, 1)	90.0	(1, 20, 18, 1)	90.0
b1l3.conv1	(12, 60, 3, 3)	(1, 30, 24, 1)	90.0	(1, 20, 20, 1)	90.0
b1l4.conv1	(12, 72, 3, 3)	(1, 30, 25, 1)	90.0	(1, 20, 20, 1)	90.0
b1l5.conv1	(12, 84, 3, 3)	(1, 15, 15, 1)	90.0	(1, 15, 15, 1)	90.0
b1l6.conv1	(12, 96, 3, 3)	(1, 15, 15, 1)	90.0	(1, 15, 15, 1)	90.0
b1l7.conv1	(12, 108, 3, 3)	(1, 15, 15, 1)	90.0	(1, 15, 15, 1)	90.0
b1l8.conv1	(12, 120, 3, 3)	(1, 15, 15, 1)	90.0	(1, 15, 15, 1)	90.0
b1l9.conv1	(12, 132, 3, 3)	(1, 15, 15, 1)	90.0	(1, 15, 15, 1)	90.0
b1l10.conv1	(12, 144, 3, 3)	(1, 25, 25, 1)	90.0	(1, 10, 10, 1)	90.0
b1l11.conv1	(12, 156, 3, 3)	(1, 25, 25, 1)	90.0	(1, 10, 10, 1)	90.0
b2l0.conv1	(12, 168, 3, 3)	(1, 30, 30, 1)	90.0	(1, 25, 25, 1)	90.0
b2l1.conv1	(12, 180, 3, 3)	(1, 30, 30, 1)	90.0	(1, 25, 25, 1)	90.0
b2l2.conv1	(12, 192, 3, 3)	(1, 30, 30, 1)	90.0	(1, 20, 20, 1)	90.0
b2l3.conv1	(12, 204, 3, 3)	(1, 30, 30, 1)	90.0	(1, 20, 20, 1)	90.0
b2l4.conv1	(12, 216, 3, 3)	(1, 15, 15, 1)	90.0	(1, 10, 10, 1)	90.0
b2l5.conv1	(12, 228, 3, 3)	(1, 15, 15, 1)	90.0	(1, 10, 10, 1)	90.0
b2l6.conv1	(12, 240, 3, 3)	(1, 15, 15, 1)	90.0	(1, 10, 10, 1)	90.0
b2l7.conv1	(12, 252, 3, 3)	(1, 15, 15, 1)	90.0	(1, 10, 10, 1)	90.0
b2l8.conv1	(12, 264, 3, 3)	(1, 15, 15, 1)	90.0	(1, 10, 10, 1)	90.0
b2l9.conv1	(12, 276, 3, 3)	(1, 20, 20, 1)	90.0	(1, 10, 10, 1)	90.0
b2l10.conv1	(12, 288, 3, 3)	(1, 20, 20, 1)	90.0	(1, 10, 10, 1)	90.0
b2l11.conv1	(12, 300, 3, 3)	(1, 20, 20, 1)	90.0	(1, 10, 10, 1)	90.0
b3l0.conv1	(12, 312, 3, 3)	(1, 30, 30, 1)	90.0	(1, 20, 20, 1)	90.0
b3l1.conv1	(12, 324, 3, 3)	(1, 30, 30, 1)	90.0	(1, 20, 20, 1)	90.0
b3l2.conv1	(12, 336, 3, 3)	(1, 30, 30, 1)	90.0	(1, 15, 15, 1)	90.0
b3l3.conv1	(12, 348, 3, 3)	(1, 30, 30, 1)	90.0	(1, 15, 15, 1)	90.0
b3l4.conv1	(12, 360, 3, 3)	(1, 25, 25, 1)	90.0	(1, 15, 15, 1)	90.0
b3l5.conv1	(12, 372, 3, 3)	(1, 25, 25, 1)	90.0	(1, 15, 15, 1)	90.0
b3l6.conv1	(12, 384, 3, 3)	(1, 25, 25, 1)	90.0	(1, 15, 15, 1)	90.0
b3l7.conv1	(12, 396, 3, 3)	(1, 25, 25, 1)	90.0	(1, 15, 15, 1)	90.0
b3l8.conv1	(12, 408, 3, 3)	(1, 25, 25, 1)	90.0	(1, 15, 15, 1)	90.0
b3l9.conv1	(12, 420, 3, 3)	(1, 25, 25, 1)	90.0	(1, 15, 15, 1)	90.0
b3l10.conv1	(12, 432, 3, 3)	(1, 25, 25, 1)	90.0	(1, 20, 20, 1)	90.0
b3l11.conv1	(12, 444, 3, 3)	(1, 25, 25, 1)	90.0	(1, 20, 20, 1)	90.0

Table 12: Layer-wise distribution of TT-Rank and sparsity ratio for the compressed ResNet-50 models on ImageNet dataset.

Layer Name	Weight Shape	Model I with 77.52% Acc.		Model II with 75.82% Acc.	
		TT Ranks	Sparsity(%)	TT Ranks	Sparsity(%)
l1.0.conv2	(64, 64, 3, 3)	(1, 36, 36, 1)	90.0	(1, 24, 24, 1)	90.0
l1.1.conv2	(64, 64, 3, 3)	(1, 36, 36, 1)	90.0	(1, 24, 24, 1)	90.0
l1.2.conv2	(64, 64, 3, 3)	(1, 36, 36, 1)	90.0	(1, 24, 24, 1)	90.0
l2.0.conv1	(128, 256, 3, 3)	(1, 48, 48, 1)	90.0	(1, 36, 36, 1)	90.0
l2.1.conv1	(128, 512, 3, 3)	(1, 48, 64, 1)	90.0	(1, 36, 36, 1)	90.0
l2.2.conv1	(128, 512, 3, 3)	(1, 48, 64, 1)	90.0	(1, 36, 36, 1)	90.0
l2.3.conv1	(128, 512, 3, 3)	(1, 48, 64, 1)	90.0	(1, 36, 36, 1)	90.0
l2.0.conv2	(128, 128, 3, 3)	(1, 48, 48, 1)	90.0	(1, 30, 30, 1)	90.0
l2.1.conv2	(128, 128, 3, 3)	(1, 48, 48, 1)	90.0	(1, 30, 30, 1)	90.0
l2.2.conv2	(128, 128, 3, 3)	(1, 48, 48, 1)	90.0	(1, 30, 30, 1)	90.0
l2.3.conv2	(128, 128, 3, 3)	(1, 48, 48, 1)	90.0	(1, 30, 30, 1)	90.0
l2.0.conv3	(512, 128, 3, 3)	(1, 72, 48, 1)	90.0	(1, 36, 36, 1)	90.0
l2.1.conv3	(512, 128, 3, 3)	(1, 72, 48, 1)	90.0	(1, 36, 36, 1)	90.0
l2.2.conv3	(512, 128, 3, 3)	(1, 72, 48, 1)	90.0	(1, 36, 36, 1)	90.0
l2.3.conv3	(512, 128, 3, 3)	(1, 72, 48, 1)	90.0	(1, 36, 36, 1)	90.0
l3.0.conv2	(256, 256, 3, 3)	(1, 64, 64, 1)	90.0	(1, 36, 36, 1)	90.0
l3.0.conv3	(1024, 256, 1, 1)	(1, 48, 48, 1)	90.0	(1, 36, 36, 1)	90.0
l3.1.conv2	(256, 256, 3, 3)	(1, 64, 64, 1)	90.0	(1, 36, 36, 1)	90.0
l3.1.conv3	(1024, 256, 1, 1)	(1, 48, 48, 1)	90.0	(1, 36, 36, 1)	90.0
l3.2.conv2	(256, 256, 3, 3)	(1, 64, 64, 1)	90.0	(1, 36, 36, 1)	90.0
l3.2.conv3	(1024, 256, 1, 1)	(1, 48, 48, 1)	90.0	(1, 36, 36, 1)	90.0
l3.3.conv2	(256, 256, 3, 3)	(1, 64, 64, 1)	90.0	(1, 36, 36, 1)	90.0
l3.3.conv3	(1024, 256, 1, 1)	(1, 48, 48, 1)	90.0	(1, 36, 36, 1)	90.0
l3.4.conv2	(256, 256, 3, 3)	(1, 64, 64, 1)	90.0	(1, 36, 36, 1)	90.0
l3.4.conv3	(1024, 256, 1, 1)	(1, 48, 48, 1)	90.0	(1, 36, 36, 1)	90.0
l3.5.conv2	(256, 256, 3, 3)	(1, 64, 64, 1)	90.0	(1, 36, 36, 1)	90.0
l3.5.conv3	(1024, 256, 1, 1)	(1, 48, 48, 1)	90.0	(1, 36, 36, 1)	90.0
l4.0.conv1	(512, 1024, 1, 1)	(1, 48, 48, 1)	90.0	(1, 48, 48, 1)	90.0
l4.0.conv2	(512, 512, 3, 3)	(1, 72, 72, 1)	90.0	(1, 48, 48, 1)	90.0
l4.0.conv3	(2048, 512, 1, 1)	(1, 64, 64, 1)	90.0	(1, 48, 48, 1)	90.0
l4.1.conv1	(512, 2048, 1, 1)	(1, 64, 64, 1)	90.0	(1, 48, 48, 1)	90.0
l4.1.conv2	(512, 512, 3, 3)	(1, 72, 72, 1)	90.0	(1, 48, 48, 1)	90.0
l4.1.conv3	(2048, 512, 1, 1)	(1, 64, 64, 1)	90.0	(1, 48, 48, 1)	90.0
l4.2.conv1	(512, 2048, 1, 1)	(1, 64, 64, 1)	90.0	(1, 48, 48, 1)	90.0
l4.2.conv2	(512, 512, 3, 3)	(1, 72, 72, 1)	90.0	(1, 48, 48, 1)	90.0
l4.2.conv3	(2048, 512, 1, 1)	(1, 64, 64, 1)	90.0	(1, 48, 48, 1)	90.0

F OTHER EXPERIMENTAL RESULTS

Compression Ratio-vs-Model Accuracy. Table 13 and 14 show the performance of compressed ResNet-20 on CIFAR-10 with different compression ratios. Here two different cases, keeping same sparsity with changing rank values and keeping same ranks with changing sparsity, are evaluated and reported.

Table 13: Experimental results on CIFAR-10 dataset with changing rank values. Here "rank ratio" represents the "individual" compression ratio **solely** brought by low-rank decomposition.

Top-1 Accuracy (%)			Params.	Rank	Sparsity
Baseline	Compressed	Δ	$\downarrow$ (%)	Ratio	
ResNet-20					
91.25	92.95	+1.70	31.3	2.0×	80%
91.25	92.47	+1.22	37.4	2.3×	80%
91.25	92.30	+1.05	44.1	2.7×	80%
91.25	92.17	+0.92	51.3	3.4×	80%
91.25	91.94	+0.69	56.2	4.1×	80%
91.25	91.78	+0.53	62.0	5.4×	80%
91.25	91.68	+0.43	67.5	7.6×	80%

Table 14: Experimental results on CIFAR-10 dataset with changing sparsity. Here "rank ratio" represents the "individual" compression ratio **solely** brought by low-rank decomposition.

Top-1 Accuracy (%)			Params.	Rank	Sparsity
Baseline	Compressed	Δ	$\downarrow$ (%)	Ratio	
ResNet-20					
91.25	93.04	+1.79	36.3	3.4×	65%
91.25	92.69	+1.44	41.4	3.4×	70%
91.25	92.31	+1.06	46.3	3.4×	75%
91.25	92.17	+0.92	51.2	3.4×	80%
91.25	92.03	+0.78	56.2	3.4×	85%
91.25	91.82	+0.57	61.3	3.4×	90%
91.25	91.77	+0.52	66.0	3.4×	95%

Changing Low-rankness and Sparsity with the Same Compression Ratio. Table 15 shows the performance of compressing ResNet-20 on CIFAR-10 with different configurations of low-rankness and sparsity under the same overall compression ratio. It is seen that, as long as keeping the overall compression ratio, changing low-rankness or sparsity has negligible impact on the accuracy, unless the sparsity is extremely high.

Table 15: Experimental results on CIFAR-10 dataset with the same overall compression ratio. Here "rank ratio" represents the "individual" compression ratio **solely** brought by low-rank decomposition.

Top-1 Accuracy (%)			Params.	Rank	Sparsity
Baseline	Compressed	Δ	$\downarrow$ (%)	Ratio	
ResNet-20					
91.25	92.03	+0.78	55.9	10.3×	65%
91.25	91.88	+0.53	55.9	6.8×	70%
91.25	91.96	+0.71	56.0	5.1×	75%
91.25	91.84	+0.59	56.2	4.1×	80%
91.25	92.12	+0.87	55.9	3.4 ×	85%
91.25	91.91	+0.70	56.6	2.9×	90%
91.25	91.53	+0.28	55.6	2.5×	95%