Co-Exploring Structured Sparsification and Low-Rank Tensor Decomposition for Compact DNNs

Yang Sui, Miao Yin, Yu Gong, Bo Yuan, *Member, IEEE*

***Abstract*—Sparsification and low-rank decomposition are two important techniques to compress deep neural network (DNN) models. To date, these two popular yet distinct approaches are typically used in separate ways; while their efficient integration for better compression performance is little explored, especially for structured sparsification and decomposition. In this paper, we perform systematic co-exploration on structured sparsification and decomposition toward compact DNN models. We first investigate and analyze several important design factors for joint structured sparsification 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 Co-Explore the benefits of structured sParsification and tensor Decomposition in an efficient way. Empirical experiments demonstrate the promising performance of our proposed solution. Notably, on the 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.**

***Index Terms*—Model Compression, Tensor Decomposition, Sparsification.**

I. 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 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 [1]–[4]. 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 [5], 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, [5] develops a singular value decomposition (SVD)-free approach to closely approximate the original DNN model via combining sparse representation and low-rank matrix factorization. Built on the interesting connection between filter decomposition and filter pruning, [6] interprets the decomposition and pruning of convolutional filters in a unified perspective. [7] combines low-rank, sparse, and quantized matrices into an additive framework with the learning-compression algorithm. Most recently, [8] proposes a collaborative 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 [5], [7] or in sequence [8], 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 [6], [8]? 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 [5]? The low-rankness/sparsity regularized

Yang Sui, Yu Gong, and Bo Yuan are with the Department of Electrical and Computer Engineering, Rutgers University, Piscataway, NJ, 08854, USA. E-mail: yang.sui@rutgers.edu, yg430@soe.rutgers.edu, bo.yuan@soe.rutgers.edu

Miao Yin is with the Department of Computer Science and Engineering at the University of Texas at Arlington, Arlington, TX 76019, USA. E-mail: miao.yin@uta.edu.

loss in [9]? 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 low-rankness and sparsity of 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 *structured* 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 *tensor* decomposition and *structured* pruning.
- We develop a unified framework that formulates the integration of *low-rank high-order tensor decomposition and structured sparsification* 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, thereby leading to efficient exploration of rich low-rankness and sparsity in the model.
- We empirically evaluate our proposed solution for various DNN models on different datasets, and the experimental results demonstrate its promising performance.

II. 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. Unstructured pruning prunes weights individually based on a certain criterion [10]–[13]. It usually brings outstanding performance but requires complicated hardware design or a dedicated sparse matrix operation library to be deployed in practice. Middle-level sparsity pruning exhibits a coarser granularity compared to unstructured pruning, encompassing N:M sparsity and block sparsity. The NVIDIA Ampere A100, equipped with sparse tensor cores, supports N:M (2:4) structured fine-grained sparsity, showcasing the promising capabilities of this intermediate level of sparsity. In N:M fine-grained sparsity, within each group of M consecutive weights in the network, at most N weights have non-zero values. The work [14] introduces a Sparse-refined Straight-Through Estimator (SR-STE) to learn the N:M sparsity from scratch. And [15] characterizes N:M sparsity as a combinatorial problem, employing learnable scores to select the desired subsets from C_M^N collections. Additionally, block sparsity represents another direction in middle-level sparsity research, targeting enhanced hardware utilization and higher sparsity levels as well. [16] clusters elements of small magnitude closer by reordering the input and output dimensions before performing pruning at a coarser granularity. [17] proposes a 1×N pruning pattern where consecutive N output kernels sharing the same input channel index are grouped into one block. This block then serves as the basic pruning granularity following an L1-norm-based filter rearrangement. Structured pruning prunes weights by selectively eliminating the entire convolutional filters or feature map channels [18]–[22], achieving practical acceleration in computational speed.

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 interpretations of DNNs, this method can be categorized into *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 [4], [23]–[26]. 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 flattening 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 [3], [27]–[33].

ADMM-based Model Compression. In recent years, ADMM has been used in several model compression papers, including single pruning [34], [35], single low-rank compression [25], joint quantization and pruning [36] and additive compressing [7], [37]. Our approach focuses on additive compression with distinct differences from [7], [37] and other ADMM-based works. This is because the low-rank approach used in [7], [37] is based on matrix decomposition, which is not the best solution for compression CNNs; while we adopt high-order tensor decomposition in our proposed additive compression. From the perspective of the ADMM process, the projection toward tensor decomposition is much more complicated than the one for matrix factorization. This is because 1) it is involved with advanced high-order tensor operation instead of straightforward 2-D matrix computation; and 2) tensor decomposition (e.g., TT) outputs multiple 4-D tensor cores; while matrix factorization only generates two 2-D matrices. Such a huge difference makes the corresponding projection on the decomposed tensor cores significantly different from the projection on low-rank used in [7], [37] or sparse matrix used in [34]–[36].

Joint Pruning and Decomposition. As observed by [5], 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, [5] decomposes the weight tensors of a pre-trained DNN model into independent low-rank and sparse parts and minimizes the reconstruction error. Similarly, [37] focuses on optimizing

the approximation error of the original model via ADMM. Different from this parallel scheme, [8], [38] adopt a sequential compression strategy via performing matrix factorization on a pruned model. In addition, [6] proposes to use the sparse/low-rank regularization term instead of reconstruction error to enforce the desired structural patterns. [39] proposes unstructured sparse and low-rank attention for transformer approximation. [7] proposes a general additive combination framework with a 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 the DNN model. However, as analyzed in Section III of our paper, high-order tensor decomposition is a more suitable low-rank method 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 a close approximation to the original model. Different from these existing works, our approach is built on structured pruning and tensor decomposition with constrained formulation as the optimization objective, demonstrating better performance.

III. 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 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 [5], 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

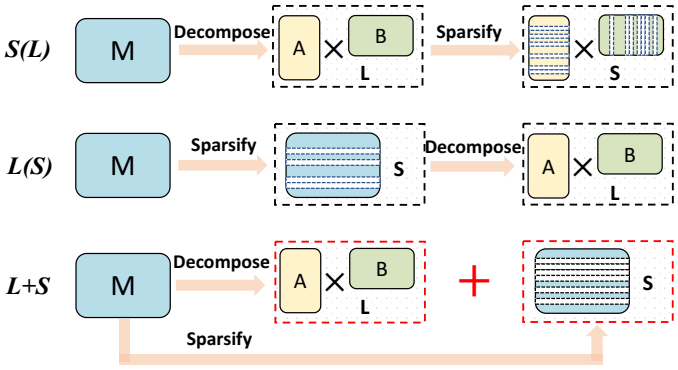


Fig. 1. Different operational sequences for joint low-rank decomposition and **structured** pruning. Here ***L*** and ***S*** represent low-rank decomposition and sparsification, respectively.

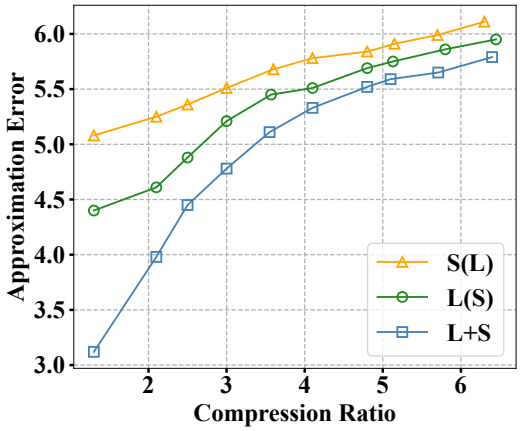


Fig. 2. The approximation error when compressing the weight tensor of one layer in ResNet-20 using different operational sequences (***L+S***, ***S(L)*** and ***L(S)***). Here mean square error (MSE) is used to measure the difference between the original uncompressed weight tensor and the reconstruction.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

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 a 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 [8], [38].

Our Proposal. Among the above-described three general operational schemes, we believe ***L+S*** is the more suitable choice when considering integrating 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 po-

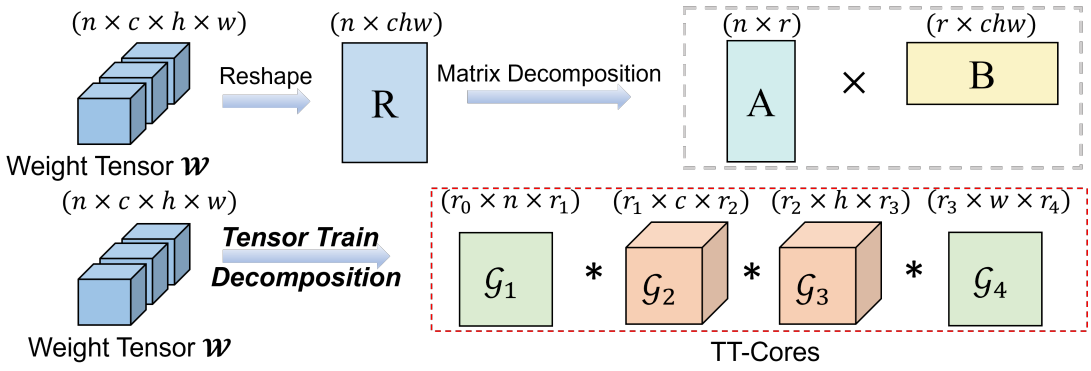


Fig. 3. Exploring low-rankness of a convolutional layer via matrix decomposition (Top) and tensor decomposition (Bottom). Here tensor train (TT) decomposition is adopted for illustration.

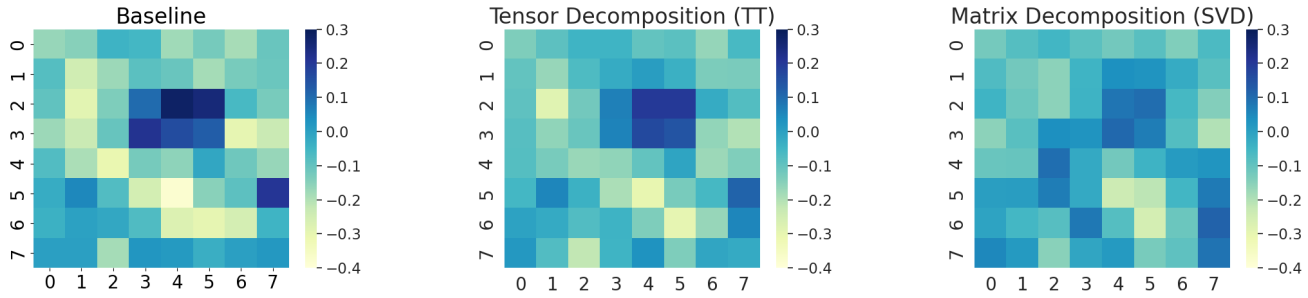


Fig. 4. Output feature maps of one layer of ResNet-20 after non-compression (Left), tensor decomposition (Middle), and matrix decomposition (Right). The visualization shown here is based on the information of one channel. **It is seen that high-order tensor decomposition makes the feature map of the compressed layer more similar to that of the original uncompressed layer.**

tential 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, 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 the DNN model in an efficient way.

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 as Figure 3. Specifically, [5] chooses the SVD-free method to factorize the DNN model and obtain the low-rank component, and [8] 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 [7], [37] are also based on 2-D matrix. Notice that though the weights of the 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 maps of the compressed convolutional layer of ResNet-20 on the CIFAR-10 dataset using different low-rank methods. For SVD, we follow the same way used in [8] to reshape the 4-D tensor weight to a 2-D weight matrix. As visualized in Figure 4, 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, the 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 ratios are reported in Appendix.

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, [5], [37] aims 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 [40]–[43] 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 a close approximation to the original model. In addition, [4], [9] explore another strategy that they consider global information via adding a regularization term to the loss function. Besides, [37], [44] 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 low-rank and sparse constraints. Our rationale lies in two observations of the drawbacks of the prior efforts. First, the approximation strategy adopted in [5], [37] 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 IV). To be specific, we report the learning curve in Section VI 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 V 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 preferred 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.

IV. Co-exploring Low-rankness and Sparsity: Our Method

Problem Formulation. Recall that the analysis in Section III 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:

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

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 $\mathrm{r}(\cdot)$ and $\mathrm{c}(\cdot)$ as well as its inherent high-order tensor format. To efficiently solve this problem, we propose to leverage the 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:

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

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:

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

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 separate parts, and independently optimize them in an iterative way.

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

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

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

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

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

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

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

Because $\widehat{\mathcal{L}}$ 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.

$$\widehat{\mathcal{L}} \leftarrow \textbf{trunc}_{\mathcal{P}}(\mathcal{L}+\mathcal{U}). \quad (8)$$

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

$$\begin{aligned}\widetilde{\mathcal{T}}((k_1,k_2),&(o_i,i_1),\cdots,(o_d,i_d))) = \\ &\mathcal{C}_0(k_1,k_2)\mathcal{C}_1(:,o_1,i_1,:)\cdots\mathcal{C}_d(:,o_d,i_d,:),\end{aligned} \quad (9)$$

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

$$\begin{aligned}\widetilde{\mathcal{T}}'((k_1,k_2),&(o_i,i_1),\cdots,(o_d,i_d))) = \\ &\mathcal{C}_0(k_1,k_2)\mathcal{C}'_1(:,o_1,i_1,:)\cdots\mathcal{C}'_d(:,o_d,i_d,:).\end{aligned} \quad (10)$$

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

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

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

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

$$\widehat{\mathcal{S}} \leftarrow \textbf{proj}_{\mathcal{Q}}(\mathcal{S}+\mathcal{V}), \quad (12)$$

where $\textbf{proj}(\cdot)$ is the projection that removes filters with the smallest ℓ_1 values. We utilize the ℓ_1 norm here as an approximation solution of its analytical solution, ℓ_2 norm, in order to save the computational cost in the optimization process. The projection is to ensure that the updated $\widehat{\mathcal{S}}$ can satisfy the structured sparse constraint.

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

$$\mathcal{U} \leftarrow \mathcal{U}+\mathcal{L}-\widehat{\mathcal{L}}, \quad \mathcal{V} \leftarrow \mathcal{V}+\mathcal{S}-\widehat{\mathcal{S}}. \quad (13)$$

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

Algorithm 1 The overall CEPD algorithm

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

```
Initialize L, L̂, S, Ŝ with W;
Initialize U := 0, V := 0
for t = 1 to T do
    Update U, V using Eq. 13;
    Update L and S using Eq. 5 and Eq. 6;
    // Update L̂ using TT-truncation
    L̂ ← trunc_P(L + U);
    // Update Ŝ using projection
    Ŝ ← proj_Q(S + V);
end for
Decompose L to TT-cores {C}_{j=0}^d;
Fine-tune model with {C}_{j=0}^d and S.
```

V. 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. We follow [6], [21], [45] to fine-tune the models on the CIFAR-10 datasets with 300 epochs and on the ImageNet datasets with 200 epochs. The entire training procedure is performed on NVIDIA-V100 GPUs with PyTorch 1.12.

Results on CIFAR-10 Dataset. Table I 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 [3], TRP [4], SVDT [9], LREL [25], ALDS [26], CaP [46] and ENC-Inf [47]), pruning-only (***S***: DCP [48], SCOP [20], FPGM [49], HRank [19], CHIP [21], CHEX [45], ISP [22]), first-pruning-then-decomposition (***L(S)***: CC [8]), and layer-wise either-pruning-or-decomposition (***S/L***: Hinge [6]).

For the ResNet-20 model, our CEPD approach results in a 0.66% accuracy increase compared to the baseline model,

TABLE I
EXPERIMENT RESULTS ON CIFAR-10. "L" DENOTES LOW-RANK DECOMPOSITION; "S" DENOTES PRUNING. IN RESNET-20, RESNET-56, AND DENSENET-40, EXPERIMENTS ARE CONDUCTED USING TWO DIFFERENT COMPRESSION RATIOS WITH VARYING SPARSITY/RANK CONFIGURATIONS. FEWER PARAMETERS AND FLOPS INDICATES SMALLER RANKS AND HIGHER SPARSITY ARE ASSIGNED, LEADING TO A FASTER AND MORE COMPACT MODEL.

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	90.50	-1.24	N/A	53.9
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**

with model size and FLOPs reductions of 56.6% and 56.2%, respectively. When employing aggressive compression, which leads to 76.4% and 68.1% reductions in model size and FLOPs, our solution still maintains high performance and surpasses ALDS in terms of accuracy, given similar model sizes and computational costs.

With respect to the DenseNet-40 model, our CEPD approach yields a 0.72% accuracy increase over the baseline model, accompanied by 41.5% and 43.0% reductions in model size and FLOPs, respectively. Upon implementing aggressive compression, resulting in 63.6% and 53.1% reductions in model size and FLOPs, our solution continues to demonstrate high performance and achieves a 0.43% higher accuracy than the baseline model, given comparable model sizes and computational costs.

In the case of another DenseNet-40 model, our CEPD approach incurs a negligible 0.02% decrease in accuracy compared to the baseline model, while still reducing model size and FLOPs by 52.6% and 50.3%, respectively.

For the MobileNetV2 model, our CEPD approach achieves a 0.45% accuracy increase compared to the baseline model, along with 48.6% and 44.2% reductions in model size and FLOPs, respectively.

A review of the presented data reveals that the CEPD approach consistently outperforms both structured pruning and low-rank decomposition methods.

Results on ImageNet Dataset. Table II summarizes the compression performance of our approach and other existing works for ResNet-18/50 on the ImageNet dataset. In addition to the previously mentioned works, we compare our CEPD with three other ***L*** works, MetaP [50], FR [51], and STABLE [33], and some ***S*** works, ABCPruner [52], Autopruner [53], WB [54], ResRep [55], FBS [56], ISP [22]. 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 a more compact model of ResNet-50, our approach can still achieve high performance – it only has a

TABLE II
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
ABCPruner	S	N/A	76.01	73.52	-2.49	92.96	91.51	-1.45	56.0	56.6
Autopruner	S	N/A	76.15	74.76	-1.39	92.87	92.15	-0.72	N/A	51.3
WB	S	N/A	76.15	75.32	-0.83	92.96	92.43	-0.53	N/A	45.6
ResRep	S	N/A	76.15	75.97	-0.18	92.87	92.75	-0.12	N/A	56.1
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**

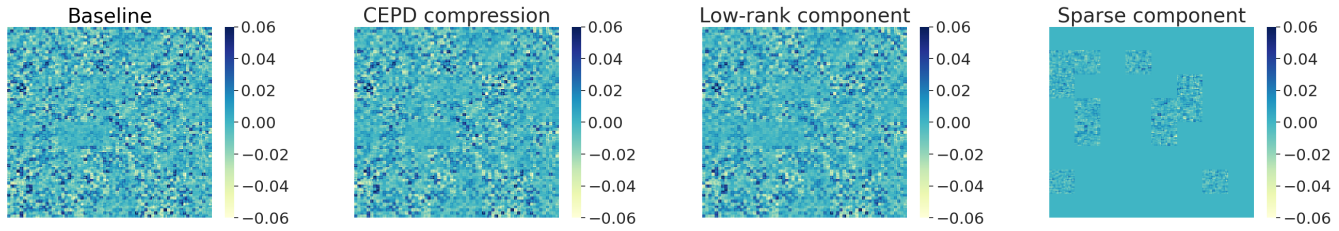


Fig. 5. Visualization of one layer of ResNet-20 before and after performing our proposed CEPD compression. Here the low-rank and sparse components of the compressed layer are also visualized. **It is seen that the low-rank component preserves most of the weight information, and some spatial patterns are contained in the sparse component.**

TABLE III
COMPARISON WITH EXISTING ADDITIVE COMPRESSION METHODS ON CIFAR-10.

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

0.31% accuracy drop with 63.3% model size reduction, which outperforms the other related works.

Comparison with Other Additive Compression Methods. Because existing ***L+S*** works ((LRSD [44] and UAF [37]) 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 III. It is seen that CEPD achieves better performance with the same or even higher compression ratio.

Inference Speedup. To showcase the practical effectiveness of our proposed approach, we assess the inference acceleration of CEPD for compressing ResNet-50 on FPGA and ASIC platforms, with diverse FLOPs reduction scenarios applied to ResNet-50. As illustrated in Table IV, the compressed models, which possess both sparsity and low-rankness, manifest remarkable gains in practical speedup. Specifically, the baseline inference time on the FPGA platform is 172.0 ms per image. By contrast, the compressed models generated by our method, which preserve 66.3%, 46.1%, and 37.1% FLOPs, respectively, demonstrate a notable practical speedup, taking 122.0 ms, 88.21 ms, and 67.72 ms per image, respectively. In contrast to the baseline model inference time of 38.10 ms per image

on the ASIC platform, the compressed models obtained from our proposed method, retaining 66.3%, 46.1%, and 37.1% of FLOPs, respectively, exhibit a substantial practical speedup. Specifically, the compressed models require only 28.26 ms, 20.88 ms, and 15.83 ms per image, respectively, to perform inference.

TABLE IV
INFERENCE TIME (PER IMAGE) FOR THE COMPRESSED RESNET-50 VIA USING CEPD.

	FLOPs Remaining	Top-1 Accuracy	FPGA Xilinx PYNQZ1	ASIC Eyeriss
Baseline	100%	76.13%	172.0ms (1×)	38.10ms (1×)
CEPD (Ours)	66.3%	77.84%	**122.0ms (1.41×)**	**28.26ms (1.35×)**
CEPD (Ours)	46.1%	77.52%	**88.21ms (1.95×)**	**20.88ms (1.82×)**
CEPD (Ours)	37.1%	75.82%	**67.72ms (2.54×)**	**15.83ms (2.41×)**

VI. IN-DEPTH ANALYSIS & ABLATION STUDY

Simultaneously Obtaining Low-rankness and Sparsity. Figure 6 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. Low-rank part loss: This loss component is derived from Eq. 7, calculated as low-rank part loss = $\frac{\lambda}{2}||\boldsymbol{\mathcal{L}}-\hat{\boldsymbol{\mathcal{L}}}+\boldsymbol{\mathcal{U}}||$. It quantifies the deviation between the low-rank representation and the sum of the low-rank part and dual variable. A lower value indicates that the imposed low-rank properties are effectively captured. Sparse loss component: Similarly, this component is also obtained from Eq. 11, calculated as sparse part loss = $\frac{\lambda}{2}||\boldsymbol{\mathcal{S}}-\hat{\boldsymbol{\mathcal{S}}}+\boldsymbol{\mathcal{V}}||$. It measures the deviation between the sparse representation and the sum of the sparse part and dual variable. A lower value signifies that the imposed sparsity properties are well captured. 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. In addition to the aforementioned low-rank loss and sparse loss, following Eq. 4 "overall loss" includes the cross-entropy loss (classification loss). Due to the additional cross-entropy loss term $f(\boldsymbol{\mathcal{L}},\boldsymbol{\mathcal{S}})$, the overall loss is larger than the low-rank/sparse part of loss.

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. We measure the Top-1 accuracy with 6 different compression ratios. Figure 7 shows the comparison results with respect to different compression ratios. It is seen that our proposed optimization procedure brings a significant accuracy increase.

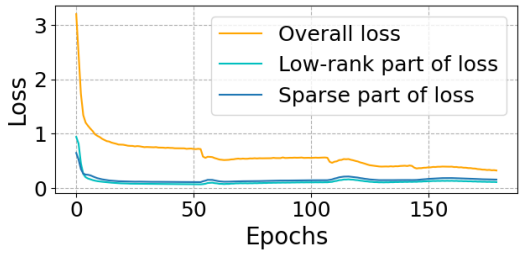


Fig. 6. Loss curves of a ResNet-20 trained on CIFAR-10 dataset using our CEPD algorithm. Here the curves of the individual low-rank and sparse loss components are also illustrated. It is seen that the low-rankness and sparsity are indeed imposed on the model via using CEPD.

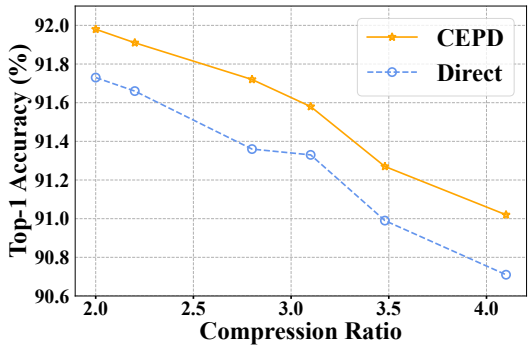


Fig. 7. The effect of an optimization procedure for jointly TT decomposing and pruning ResNet-20 on CIFAR10. Here CEPD and the direct method use the same TT-rank setting and sparsity ratio. However, the direct method does not perform optimization on the original model. Instead, it first performs TT decomposition and then prunes the difference between the 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 in the same way that CEPD uses. It is seen that our proposed optimization procedure in CEPD brings significant accuracy increase.

Visualization. Figure 5 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 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.

More results for analysis in Question #1. Table V, VI 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 on CIFAR-10 and ResNet-50 on ImageNet, and the compression ratio is set as 3.0 for all the layers. "l" and "b" in the Layer Name denotes the stage and block, respectively. For instance, "l1b1" denotes the layer in the first stage and the first block of the ResNet-20 mode. The weight shape is (Input channel, Output channel, Kernel size, and Kernel size). It is seen that ***L+S*** scheme always brings the smallest approximation errors. For example, in the "l2b1.conv1" layer, ***L+S*** only incurs the 2.44 approximation error, which is less than that of ***L(S)***, 2.52 and ***S(L)***, 2.74.

More results for analysis in Question #2. Table VII shows

TABLE V
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

the difference (in terms of mean square error (MSE)) between the output feature maps of the original layer and the compressed one in ResNet-20 on the CIFAR-10 dataset. Here SVD and tensor train (TT) are adopted for matrix decomposition and tensor decomposition, respectively. Upon compressing the "l2b1.conv2" layer utilizing SVD and TT techniques with a compression ratio of 1.91 and 2.03, respectively, the resulting approximation errors are measured at 15.40 and 10.13. It shows that with the same or even higher compression ratio, high-order tensor decomposition always brings a smaller approximation error than matrix decomposition.

Table VIII shows the difference between the output feature maps of one original layer and the compressed version in ResNet-20 on the CIFAR-10 dataset and the difference in the final accuracy. Here SVD, Tucker, and TT are adopted for low-rank decompositions. Upon applying compression to the "l3b1.conv2" layer using SVD, Tucker, and TT methods with a compression ratio of 1.98, 1.97, and 2.01, respectively, the corresponding approximation errors are evaluated as 5.60, 4.80, and 4.16. Additionally, the resultant accuracy decreases are determined as 3.0, 2.1, and 0.8, respectively. It is seen that high-order tensor decomposition always brings smaller approximation errors than the matrix decomposition and TT always has the least impact on the final accuracy.

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

Changing Low-rankness and Sparsity with the Same Compression Ratio. Table XI shows the performance of compressing ResNet-20 on CIFAR-10 with different configurations of low-rankness and sparsity under the same overall compression ratio. Notice there is a fluctuation in top-1 accuracy with increasing sparsity. This observation hints at a potential sweet spot in the trade-off between the "low-rank part" and the

TABLE VI
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 VII
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

"sparse part" where the rank and sparsity are determined by its natural inherent structure. Specifically, first, the inherent rank of the weight matrix itself is a primary factor. If the weights naturally clusters into smaller dimensions, the rank will be lower. Second, outliers in weights can significantly affect the estimated rank and sparsity. The method considering both low-rank and sparsity is designed to handle outliers by isolating them into the sparse component, and the effectiveness of this

TABLE VIII
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

TABLE IX
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	↓ (%)	Ratio	
ResNet-20					
91.25	92.19	+0.94	31.3	2.0×	80%
91.25	92.06	+0.81	37.4	2.3×	80%
91.25	92.02	+0.77	44.1	2.7×	80%
91.25	91.97	+0.72	51.3	3.4×	80%
91.25	91.83	+0.58	56.2	4.1×	80%
91.25	91.69	+0.44	62.0	5.4×	80%
91.25	91.56	+0.31	67.5	7.6×	80%

separation can influence the resulting rank. Exploring how to determine this "sweet spot" might be a promising research direction. Fig. 8 summaries the relationship between accuracy and varying compression ranks and sparsity levels in Table IX, X, and XI. The trend demonstrates that reduced sparsity and increased ranks contribute to higher accuracy in most cases.

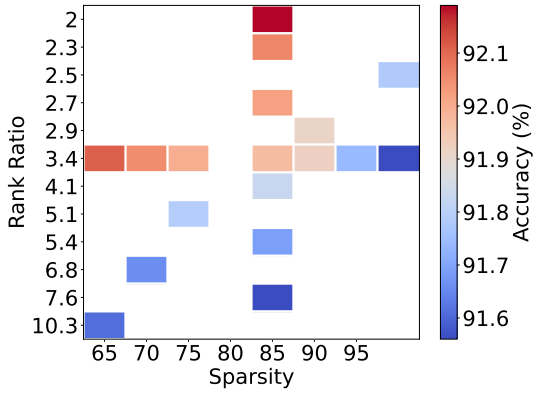


Fig. 8. The relationship between model accuracy and different sparsity/rank configurations shown in Table IX, X, and XI. A box displaying more red or blue denotes higher or lower accuracy, respectively. The trend illustrates that, in most cases, decreasing sparsity and increasing ranks lead to higher accuracy.

Compression Cost. We also exhibit the training cost of ResNet-50 on the ImageNet dataset in Table XII. Given that our method requires a training stage for decomposition and pruning before fine-tuning, in line with regularization-based works [57], additional computational cost is required than one-shot pruning work [10]. Suppose the FLOPs of the forward propagation is f_F, the FLOPs of the backward propagation would be $2f_F$. According to Eq. 5 and Eq. 6, our optimization method incurs about three more addition/subtraction operations on the overall parameters than the standard SGD. Therefore, the pruning FLOPs are calculated as $(3f_F + 3P) \times Epochs \times T$ where P is the total number of parameters, and T is the number of total training samples in one epoch. Notice that the FLOPs of calculating the multipliers in Eq. 13 and projection functions in Eq. 8 and Eq. 12 are ignored here since they only occur once per epoch in our implementation, while $\boldsymbol{\mathcal{L}}$ and $\boldsymbol{\mathcal{S}}$ are calculated per iteration. Considering the number of iterations in one epoch is typically large, the FLOPs contribution to calculating the multipliers and projection functions is negligible. For the fine-tuning phase FLOPs, we calculate it as $3f_F \times FR$, where FR denotes the FLOPs remaining percentage. Total cost is calculated by integrating the pruning phase FLOPs and fine-tuning phase FLOPs. From Table XII, our method can perform a similar training cost on the compression stage to the regularization-based method [57] with a higher top-1 accuracy.

On the other hand, although our method outperforms the regularization-based approach, it falls short in speed when compared to one-shot pruning, whose pruning phase cost is nearly negligible. Our strength lies in performance rather than compression cost. Balancing both compression speed and final performance presents a promising direction for future research.

TABLE X
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	↓ (%)	Ratio	
ResNet-20					
91.25	92.11	+0.86	36.3	3.4×	65%
91.25	92.05	+0.80	41.4	3.4×	70%
91.25	92.00	+0.75	46.3	3.4×	75%
91.25	91.97	+0.72	51.3	3.4×	80%
91.25	91.92	+0.67	56.2	3.4×	85%
91.25	91.74	+0.49	61.3	3.4×	90%
91.25	91.56	+0.31	66.0	3.4×	95%

TABLE XI
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	↓ (%)	Ratio	
ResNet-20					
91.25	91.61	+0.36	55.9	10.3×	65%
91.25	91.66	+0.41	55.9	6.8×	70%
91.25	91.79	+0.54	56.0	5.1×	75%
91.25	91.83	+0.58	56.2	4.1×	80%
91.25	91.90	+0.65	55.9	3.4×	85%
91.25	91.91	+0.66	56.6	2.9×	90%
91.25	91.78	+0.53	55.6	2.5×	95%

TABLE XII
TRAINING TIME FOR COMPRESSING RESNET-50 OF OUR PROPOSED METHOD. * DENOTES THE REPRODUCTION RESULT. T IS THE NUMBER OF TOTAL TRAINING SAMPLES IN ONE EPOCH.

	FLOPs Remaining	Top-1 Accuracy	Pruning Phase Cost (B)	Fine-tuning Phase Cost (B)	Total Cost (B)	Total Cost Saving
Baseline* [57]	51.4%	75.67%	12.3×120T	6.27×200T	2730T	1×
CEPD (Ours)	46.1%	**77.52%**	12.4×120T	5.66×200T	2620T	1.04×
CEPD (Ours)	37.1%	**76.13%**	12.4×120T	4.55×200T	2398T	1.14×

VII. Conclusion

In this paper, we propose to systematically co-explore the tensor-based low-rankness and structured sparsity for efficient model compression. By performing a comprehensive analysis of 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.

Acknowledgments

This work was partially supported by National Science Foundation under CCF-1955909 and CMMI-2152908.

References

[1] Z. Wang, C. Li, and X. Wang, "Convolutional neural network pruning with structural redundancy reduction," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 14 913–14 922.

[2] S. Gao, F. Huang, W. Cai, and H. Huang, "Network pruning via performance maximization," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 9270–9280.

[3] N. Li, Y. Pan, Y. Chen, Z. Ding, D. Zhao, and Z. Xu, "Heuristic rank selection with progressively searching tensor ring network," *Complex & Intelligent Systems*, pp. 1–15, 2021.

[4] Y. Xu, Y. Li, S. Zhang, W. Wen, B. Wang, Y. Qi, Y. Chen, W. Lin, and H. Xiong, "Trp: Trained rank pruning for efficient deep neural networks," in *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI-20*, 2020, pp. 977–983.

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

[6] Y. Li, S. Gu, C. Mayer, L. V. Gool, and R. 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*, 2020, pp. 8018–8027.

[7] Y. Idelbayev and M. 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*. Springer, 2021, pp. 233–248.

[8] Y. Li, S. Lin, J. Liu, Q. Ye, M. Wang, F. Chao, F. Yang, J. Ma, Q. Tian, and R. Ji, "Towards compact cnns via collaborative compression," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 6438–6447.

[9] H. Yang, M. Tang, W. Wen, F. Yan, D. Hu, A. Li, H. Li, and Y. 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*, 2020, pp. 678–679.

[10] S. Han, H. Mao, and W. J. Dally, "Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding," *arXiv preprint arXiv:1510.00149*, 2015.

[11] Y.-J. Zheng, S.-B. Chen, C. H. Ding, and B. Luo, "Model compression based on differentiable network channel pruning," *IEEE Transactions on Neural Networks and Learning Systems*, 2022.

[12] Y. He, P. Liu, L. Zhu, and Y. Yang, "Filter pruning by switching to neighboring cnns with good attributes," *IEEE Transactions on Neural Networks and Learning Systems*, 2022.

[13] M. Lin, L. Cao, S. Li, Q. Ye, Y. Tian, J. Liu, Q. Tian, and R. Ji, "Filter sketch for network pruning," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 33, no. 12, pp. 7091–7100, 2021.

[14] A. Zhou, Y. Ma, J. Zhu, J. Liu, Z. Zhang, K. Yuan, W. Sun, and H. Li, "Learning n: M fine-grained structured sparse neural networks from scratch," in *International Conference on Learning Representations*, 2020.

[15] Y. Zhang, M. Lin, Z. Lin, Y. Luo, K. Li, F. Chao, Y. Wu, and R. Ji, "Learning best combination for efficient n: M sparsity," *Advances in Neural Information Processing Systems*, vol. 35, pp. 941–953, 2022.

[16] Y. Ji, L. Liang, L. Deng, Y. Zhang, Y. Zhang, and Y. Xie, "Tetris: Tile-matching the tremendous irregular sparsity," *Advances in neural information processing systems*, vol. 31, 2018.

[17] M. Lin, Y. Zhang, Y. Li, B. Chen, F. Chao, M. Wang, S. Li, Y. Tian, and R. Ji, "1xn pattern for pruning convolutional neural networks," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 45, no. 4, pp. 3999–4008, 2022.

[18] L. Liebenwein, C. Baykal, H. Lang, D. Feldman, and D. Rus, "Provable filter pruning for efficient neural networks," *arXiv preprint arXiv:1911.07412*, 2019.

[19] M. Lin, R. Ji, Y. Wang, Y. Zhang, B. Zhang, Y. Tian, and L. Shao, "Hrank: Filter pruning using high-rank feature map," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 1529–1538.

[20] Y. Tang, Y. Wang, Y. Xu, D. Tao, C. XU, C. Xu, and C. Xu, "Scop: Scientific control for reliable neural network pruning," in *Advances in Neural Information Processing Systems*, vol. 33, 2020, pp. 10 936–10 947.

[21] Y. Sui, M. Yin, Y. Xie, H. Phan, S. Aliari Zonouz, and B. Yuan, "Chip: Channel independence-based pruning for compact neural networks," *Advances in Neural Information Processing Systems*, vol. 34, 2021.

[22] A. Ganjdanesh, S. Gao, and H. Huang, "Interpretations steered network pruning via amortized inferred saliency maps," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2022.

[23] C. Tai, T. Xiao, Y. Zhang, X. Wang *et al.*, "Convolutional neural networks with low-rank regularization," in *International Conference on Learning Representations*, 2016.

[24] C. Li and C. Shi, "Constrained optimization based low-rank approximation of deep neural networks," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 732–747.

[25] Y. Idelbayev and M. 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*, 2020, pp. 8049–8059.

[26] L. Liebenwein, A. Maalouf, D. Feldman, and D. Rus, "Compressing neural networks: Towards determining the optimal layer-wise decomposition," *Advances in Neural Information Processing Systems*, vol. 34, 2021.

[27] E. L. Denton, W. Zaremba, J. Bruna, Y. LeCun, and R. Fergus, "Exploiting linear structure within convolutional networks for efficient evaluation," in *Advances in neural information processing systems*, 2014, pp. 1269–1277.

[28] Y.-D. Kim, E. Park, S. Yoo, T. Choi, L. Yang, and D. Shin, "Compression of deep convolutional neural networks for fast and low power mobile applications," in *International Conference on Learning Representations*, 2016.

[29] A. Novikov, D. Podoprikhin, A. Osokin, and D. P. Vetrov, "Tensorizing neural networks," *Advances in Neural Information Processing Systems*, vol. 28, pp. 442–450, 2015.

[30] W. Wang, Y. Sun, B. Eriksson, W. Wang, and V. Aggarwal, "Wide compression: Tensor ring nets," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2018, pp. 9329–9338.

[31] J. Kossaifi, A. Bulat, G. Tzimiropoulos, and M. 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*, 2019, pp. 7822–7831.

[32] J. Kossaifi, A. Toisoul, A. Bulat, Y. Panagakis, T. M. Hospedales, and M. 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*, 2020, pp. 6060–6069.

[33] A.-H. Phan, K. Sobolev, K. Sozykin, D. Ermilov, J. Gusak, P. Tichavskỳ, V. Glukhov, I. Oseledets, and A. Cichocki, "Stable low-rank tensor decomposition for compression of convolutional neural network," in *European Conference on Computer Vision*. Springer, 2020, pp. 522–539.

[34] T. Zhang, S. Ye, K. Zhang, J. Tang, W. Wen, M. Fardad, and Y. Wang, "A systematic dnn weight pruning framework using alternating direction method of multipliers," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 184–199.

[35] M. A. Carreira-Perpinán and Y. Idelbayev, ""learning-compression" algorithms for neural net pruning," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 8532–8541.

[36] H. Yang, S. Gui, Y. Zhu, and J. Liu, "Automatic neural network compression by sparsity-quantization joint learning: A constrained optimization-based approach," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 2178–2188.

[37] Y. Ma, R. Chen, W. Li, F. Shang, W. Yu, M. Cho, and B. Yu, "A unified approximation framework for compressing and accelerating deep neural networks," in *2019 IEEE 31st International Conference on Tools with Artificial Intelligence (ICTAI)*. IEEE, 2019, pp. 376–383.

[38] A. Dubey, M. Chatterjee, and N. Ahuja, "Coreset-based neural network compression," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 454–470.

[39] B. Chen, T. Dao, E. Winsor, Z. Song, A. Rudra, and C. Ré, "Scatterbrain: Unifying sparse and low-rank attention," *Advances in Neural Information Processing Systems*, vol. 34, pp. 17 413–17 426, 2021.

[40] T. Bouwmans, N. S. Aybat, and E.-h. Zahzah, *Handbook of robust low-rank and sparse matrix decomposition: Applications in image and video processing*. CRC Press, 2016.

[41] E. Richard, B. Francis, and J.-P. Vert, "Intersecting singularities for multi-structured estimation," in *International Conference on Machine Learning*. PMLR, 2013, pp. 1157–1165.

[42] E. Richard, P.-A. Savalle, and N. Vayatis, "Estimation of simultaneously sparse and low rank matrices," *arXiv preprint arXiv:1206.6474*, 2012.

[43] X. Luo, "Recovering model structures from large low rank and sparse covariance matrix estimation," *arXiv preprint arXiv:1111.1133*, 2011.

[44] K. Guo, X. Xie, X. Xu, and X. Xing, "Compressing by learning in a low-rank and sparse decomposition form," *IEEE Access*, vol. 7, pp. 150 823–150 832, 2019.

[45] Z. Hou, M. Qin, F. Sun, X. Ma, K. Yuan, Y. Xu, Y.-K. Chen, R. Jin, Y. Xie, and S.-Y. Kung, "Chex: Channel exploration for cnn model compression," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 12 287–12 298.

[46] B. Minnehan and A. Savakis, "Cascaded projection: End-to-end network compression and acceleration," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 10 715–10 724.

[47] H. Kim, M. U. K. Khan, and C.-M. Kyung, "Efficient neural network compression," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 12 569–12 577.

[48] Z. Zhuang, M. Tan, B. Zhuang, J. Liu, Y. Guo, Q. Wu, J. Huang, and J. Zhu, "Discrimination-aware channel pruning for deep neural networks," in *Advances in Neural Information Processing Systems*, 2018, pp. 875–886.

[49] Y. He, P. Liu, Z. Wang, Z. Hu, and Y. Yang, "Filter pruning via geometric median for deep convolutional neural networks acceleration," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 4340–4349.

[50] Z. Liu, H. Mu, X. Zhang, Z. Guo, X. Yang, K.-T. Cheng, and J. Sun, "Metapruning: Meta learning for automatic neural network channel pruning," in *Proceedings of the IEEE International Conference on Computer Vision*, 2019, pp. 3296–3305.

[51] B.-S. Chu and C.-R. Lee, "Low-rank tensor decomposition for compression of convolutional neural networks using funnel regularization," *arXiv preprint arXiv:2112.03690*, 2021.

[52] M. Lin, R. Ji, Y. Zhang, B. Zhang, Y. Wu, and Y. Tian, "Channel pruning via automatic structure search," in *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, 2020, pp. 673 – 679.

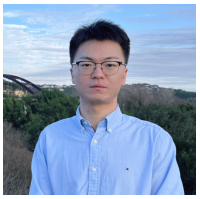
[53] J.-H. Luo and J. Wu, "Autopruner: An end-to-end trainable filter pruning method for efficient deep model inference," *Pattern Recognition*, vol. 107, p. 107461, 2020.

[54] Y. Zhang, M. Lin, C.-W. Lin, J. Chen, Y. Wu, Y. Tian, and R. Ji, "Carrying out cnn channel pruning in a white box," *IEEE Transactions on Neural Networks and Learning Systems*, 2022.

[55] X. Ding, T. Hao, J. Tan, J. Liu, J. Han, Y. Guo, and G. Ding, "Resrep: Lossless cnn pruning via decoupling remembering and forgetting," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 4510–4520.

[56] X. Gao, Y. Zhao, Łukasz Dudziak, R. Mullins, and C. zhong Xu, "Dynamic channel pruning: Feature boosting and suppression," in *International Conference on Learning Representations*, 2019.

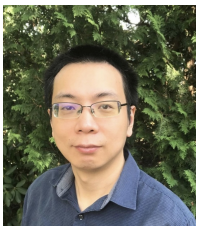
[57] W. Wen, C. Wu, Y. Wang, Y. Chen, and H. Li, "Learning structured sparsity in deep neural networks," *Advances in neural information processing systems*, vol. 29, pp. 2074–2082, 2016.



Yang Sui is a Ph.D. candidate in the Department of Electrical and Computer Engineering at Rutgers University. His research focuses on Efficient AI, including model compression and efficient training for various AI tasks, such as image classification and generative models. The primary goal of his research is to minimize the storage and computational requirements of current large-scale AI models through compression techniques, including pruning, low-rank decomposition, and quantization.

Miao Yin is an Assistant Professor in the Department of Computer Science and Engineering at the University of Texas at Arlington. He received his Ph.D. with the Graduate Academic Achievement Award from Rutgers University. His research is focused on algorithm-hardware co-optimization for high-performance AI systems. Dr. Yin serves as the technical committee member of the IEEE/ACM International Conference on Computer-Aided Design (ICCAD), the International Symposium on Quality Electronic Design (ISQED), and the IEEE International Conference on Computer Design (ICCD).

Yu Gong is a Ph.D. student in the Department of Electrical and Computer Engineering at Rutgers University. His research focuses on VLSI, computer architecture, and algorithm-hardware co-design, aiming to optimize high-performance AI and signal processing systems. His work has been published in top-tier journals and conferences such as ISCA, FPGA, and Transactions on Computer.



Bo Yuan received bachelor and master degrees from Nanjing University, China, and he received PhD degree from Department of Electrical and Computer Engineering at University of Minnesota, Twin Cities. He is currently the associate professor of Department of Electrical and Computer Engineering in Rutgers University. He has broad research interests in all the technical stacks towards building next generation intelligent systems, including efficient AI/ML algorithm and hardware, efficient and intelligent robotic computing, real-time energy-efficient wireless, secure/reliable/robust AI, AI-powered domain applications (e.g., smart manufacturing and transportation), etc. He is the recipient of NSF CAREER award, IEEE TCSDM Early-Career Award and Broadcom Global Research Competition Finalist Award. Dr. Yuan serves as technical committee track chair and (senior) technical committee member for several IEEE/ACM conferences. He is the associate editor of Springer Journal of Signal Processing System.