Unified Framework for Pre-trained Neural Network Compression via Decomposition and Optimized Rank Selection

Ali Aghababaei-Harandi and Massih-Reza Amini

Université Grenoble Alpes, CNRS,
Computer Science Laboratory LIG, Grenoble, France
{Firstname.Lastname}@univ-grenoble-alpes.fr

Abstract. Despite their high accuracy, complex neural networks demand significant computational resources, posing challenges for deployment on resource constrained devices such as mobile phones and embedded systems. Compression algorithms have been developed to address these challenges by reducing model size and computational demands while maintaining accuracy. Among these approaches, factorization methods based on tensor decomposition are theoretically sound and effective. However, they face difficulties in selecting the appropriate rank for decomposition. This paper tackles this issue by presenting a unified framework that simultaneously applies decomposition and rank selection, employing a composite compression loss within defined rank constraints. Our method includes an automatic rank search in a continuous space, efficiently identifying optimal rank configurations for the pre-trained model by eliminating the need for additional training data and reducing computational overhead in the search step. Combined with a subsequent fine-tuning step, our approach maintains the performance of highly compressed models on par with their original counterparts. Using various benchmark datasets and models, we demonstrate the efficacy of our method through a comprehensive analysis.

Keywords: Neural Network, Decomposition, Optimal Rank.

1 Introduction

In recent years, deep learning has revolutionized various scientific fields, including computer vision and natural language processing [26]. Complex neural networks with millions or billions of parameters have achieved unprecedented accuracy. However, their size poses challenges for deployment on resource-limited devices like mobile phones and edge devices [27]. The storage, memory, and processing requirements of these models often prove to be unfeasible or excessively costly, thus limiting their practicality and accessibility.

Recent research has introduced various compression algorithms to address cost-effectiveness, scalability, and real-time responsiveness [22]. These approaches,

which reduce a model's size and computational demands while preserving accuracy, can be classified into four primary categories. One straightforward method is *pruning*, which involves removing insignificant weights from the model [4]. *Quantization* reduces the precision of numerical values, typically transitioning from 32-bit floating-point numbers to lower bit-width fixed-point numbers [24]. *Knowledge distillation* trains a smaller "student" model to mimic a larger "teacher" model, resulting in a compact model with similar performance [3]. Lastly, *low-rank factorization* decomposes weight matrices or tensors into smaller components, reducing the number of parameters [2, 35, 36]. While effective, selecting the appropriate rank for decomposition remains a significant challenge.

Non-uniqueness in tensor rank is a major challenge in tensor decomposition research. Most tensor decomposition problems, especially CP decomposition, are NP-hard [13], and allow different decompositions of a same tensor even though some works tries to approximating the ranks of a tensor in a practical way [34, 11]. Finding the ideal rank is an ongoing research topic, and determining multiple tensor ranks for deep neural network layers is not suitable for conventional hyperparameter selection methods like cross-validation. Typically, a single rank is chosen for the decomposition of layers based on a compression rate, but this can lead to significant performance degradation in complex models.

Recent studies propose automated methods for determining tensor decomposition ranks [5, 20, 33]. However, these approaches, including reinforcement learning, greedy search algorithms, and SuperNet search, can be computationally expensive and time-consuming, especially for large models and datasets. Their effectiveness often depends on hyperparameters like learning rates or regularization parameters, which are challenging to tune. Additionally, existing methods do not cover a wide enough search space to achieve ideal compression rates.

This paper introduces a unified framework that simultaneously addresses tensor decomposition and optimal rank selection using a composite compression loss within specified rank constraints. Also, when we combine this rank search with a subsequent fine-tuning step, our experiments show that the highly compressed model performs similarly to the original model. The key contributions of this paper are:

- Our proposed method allows to achieve maximum compression rates by covering all ranks in the search space through a simple and efficient multi-step search process that explores ranks from low to high resolution.
- The proposed search method involves an automatic rank search in a continuous space, which efficiently identifies the optimal rank configurations for layer decomposition without requiring training data.
- We perform a comprehensive analysis of the various components of our approach, highlighting its efficacy across various benchmark datasets and models such as convolution and transformer-based models. we achieved improvement in some experiments specifically improvement in all metrics in the case of ResNet-18, while in another experiment we had competitive results. Moreover, our method speeds up the search phase compared to other related work.

2 Related Work

Low-rank factorization techniques, particularly tensor decomposition, have gained attention in deep learning, especially in natural language processing (NLP) [22]. These methods provide an efficient means of fine-tuning large language models, offering advantages over alternative techniques such as quantization [24], knowledge distillation [3], and gradient-based pruning [37]. In this paper, we focus on tensor decomposition, which proved to be a robust compression tool with a high compression rate and a relatively lower computational cost. Their applications extend beyond NLP and have also been applied in computer vision [36]. However, selecting the appropriate rank for compressing deep neural models using decomposition techniques is NP-hard [13]. Research in this area falls into two main approaches.

The first approach relies on a rank-fixed setting, where the ranks of layers are determined based on a predefined compression rate target. Some work used a low-rank loss to substitute the weights of convolution layers with their low-rank approximations [38]. The two main low-rank approximation methods applied on pre-trained models are CP and Tucker decomposition [16]. Recent studies have revealed that fine-tuning after CP decomposition can be unstable and have addressed this issue by integrating a stability term into the decomposition process [25]. In addition, some work decomposed convolution and fully connected layers with tensor train, and trained the model from scratch [22]. However, tensor decomposition in a fixed-rank setting presents certain challenges. First, selecting the appropriate rank for different layers is complex and often relies on human expertise. Second, there is a lack of interpretable patterns between layer ranks, leading to inconsistencies among the chosen ranks between layers. Furthermore, the fixed rank strategy overlooks the varying importance of layers [19], which can result in suboptimal approximations that can lead to accuracy drops or insufficient compression rates.

The second approach involves determining the optimal ranks by setting the optimization problem on the basis of the ranks of layers. One technique consists in iteratively decreasing the ranks of the layers at each step of the search phase [12]. The discrete nature of rank search lends itself to discrete search algorithms, such as reinforcement learning and progressive search, to identify optimal ranks [20]. Other methods impose constraints on ranks and budget, using iterative optimization strategies [37]. More recent studies explore continuous search spaces to determine optimal ranks [8, 33, 32, 6].

To address time complexity issues, these approaches depend on training data to search for ranks, restricting exploration to a limited search space, and thereby limiting the achievable compression rate. In contrast, we introduce a novel optimization problem that minimizes a decomposition loss while enforcing a rank loss constraint independent of the training data, which accelerates the search process for large models. For rank selection, we propose an efficient dichotomous search method that is both fast and allows for a broader range of rank exploration, ultimately enhancing the compression rate.

3 Background and Preliminaries

In the following, we represent indices using italicized letters and sets with italic calligraphic letters. For two-dimensional arrays (matrices) and one-dimensional arrays (vectors), we use bold capital letters and bold lowercase letters, respectively. Finally, tensors are represented as multidimensional arrays with bold calligraphic capital letters.

A fundamental technique for efficiently representing and processing tensors is tensor decomposition. This technique transforms a multidimensional array of data into a series of lower-dimensional tensors, thereby reducing both the representation size and computational complexity. The prevalent tensor decomposition techniques encompass canonical polyadic (CP) [10], Tucker [29], tensor train (TT) [23], and tensor ring (TR) decomposition [22].

In our work, we employ both the TT and CP decompositions. TT decomposition supports fast multilinear multiplication and integration while preserving structure, and CP decomposition has been shown to achieve high parameter reduction in CNNs with small performance drops [22]. In the following, we present the TT decomposition due to its structural advantages in capturing complex dependencies.

TT decomposition decomposes a tensor into smaller tensors with dimensions connected as a chain to each other. This decomposition mathematically can be represented as follows:

$$\hat{\boldsymbol{\mathcal{W}}}^{(R_1,\ldots,R_{N-1})}(i_1,i_2,...,i_N) = \sum_{j_1=1}^{R_1}\cdots\sum_{j_{N-1}=1}^{R_{N-1}} \boldsymbol{\mathcal{G}}_1(i_1,j_1)\boldsymbol{\mathcal{G}}_2(j_1,i_2,j_2) \cdots \boldsymbol{\mathcal{G}}_N(j_{N-1},i_N), \tag{1}$$

where the tuple $(R_1, R_2, \ldots, R_{N-1})$ represents the rank of the TT decomposition, and $\mathcal{G}_k$ are the TT cores with sizes $R_{k-1} \times I_k \times R_k$, and $R_0 = R_N = 1$. For a given convolutional layer with a weight tensor $\boldsymbol{\mathcal{W}} \in R^{b\times h\times w\times c}$, the forward process for an input tensor $\boldsymbol{\mathcal{X}} \in R^{k_1\times k_2\times k_3}$ can be expressed as:

$$\boldsymbol{\mathcal{Y}} = \sum_{i_1=0}^{k_1-1}\sum_{i_2=0}^{k_2-1}\sum_{i_3=0}^{k_3-1} \boldsymbol{\mathcal{W}}(t,x+i_1,y+i_2,z+i_3)\boldsymbol{\mathcal{X}}(i_1,i_2,i_3). \tag{2}$$

Specifically, we investigate how the weight tensor of a convolutional layer can be decomposed into multiple smaller convolution operations. We utilize TT decomposition, as detailed in the following formulations:

$$\boldsymbol{\mathcal{Y}} = \sum_{r_1=1}^{R_1}\sum_{r_2=1}^{R_2} \boldsymbol{\mathcal{G}}_t(t,r_1)\Bigg(\sum_{i_1=0}^{k_1-1}\sum_{i_2=0}^{k_2-1} \boldsymbol{\mathcal{G}}_y(r_1,x+i_1,y+i_2,r_2) \Bigg(\sum_{i_3=0}^{k_3-1} \boldsymbol{\mathcal{G}}_s(i_3,r_2)\boldsymbol{\mathcal{X}}(i_1,i_2,i_3)\Bigg)\Bigg). \tag{3}$$

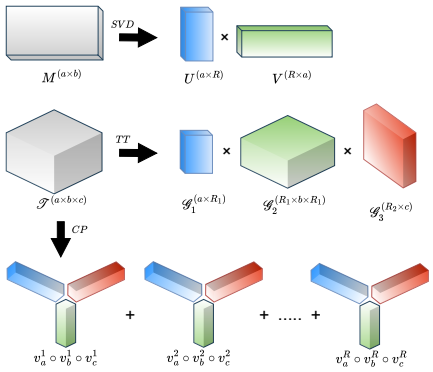

Fig. 1: An illustration of matrix decomposition (**upper row**) using SVD for a matrix $M \in \mathbb{R}^{a \times b}$, alongside Tensor Train decomposition (**middle row**), and CP decomposition (**bottom row**) for a tensor $\boldsymbol{\mathcal{T}} \in \mathbb{R}^{a \times b \times c}$.

The CP decomposition expresses a multi-dimensional tensor into a sum of rank-one tensors. It follows a well-established factorization process that has been extensively studied in prior works [10]. Figure 1 illustrates TT decomposition and CP decomposition in relation to matrix decomposition using SVD.

4 Optimal Rank Tensor Decomposition

The proposed method, denoted as Rank adapt tENsor dEcomposition (`RENE`) and illustrated in Figure 2, involves tensor decomposition with an automatic search for optimal ranks. The approach begins with a pre-trained neural network and aims to decompose its weight tensors layer by layer into lower-rank approximations while minimizing both decomposition and rank losses. This is achieved through an iterative optimization process that updates the decomposition weights and rank coefficients.

At each layer $i \in \{1, \ldots, n\}$, rank coefficients $(p_j^i)_j$ related to a set of ranks $\mathcal{R}_i$ for decomposition (Figure 2 (left)) are found iteratively and progressively refined until a single optimal rank is determined. The decomposed network with this optimal rank is fine-tuned to align its outputs with the original model (Figure 2 (right)), ensuring that the compressed model retains the performance of the original while being more efficient.

Equations (1) and (3) show that both the number of parameters and computation complexity are directly proportional to the rank of the layer. Consequently, selecting a lower rank results in a reduction in these computational costs. From this observation, we define the decomposition problem as the minimization of a decomposition error under a rank constraint.

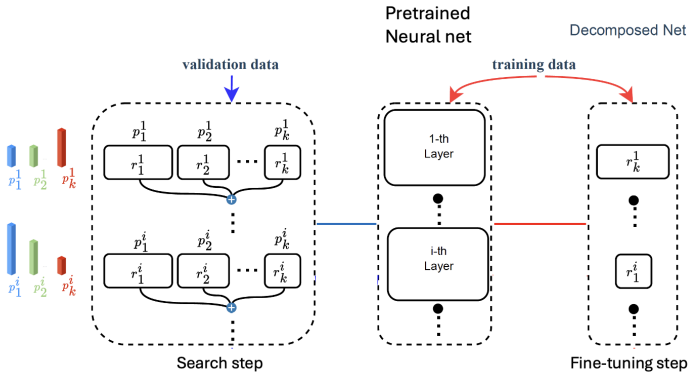

Fig. 2: Overview of RENE: Starting with a pre-trained neural network, weight tensors are decomposed layer by layer into lower-rank approximations. Rank coefficients for each layer are refined until optimal (left), followed by fine-tuning of the decomposed network (right).

4.1 Problem Formulation

Given a pre-trained neural network with n hidden layers and weights $\{\boldsymbol{\mathcal{W}}_i\}_{i=1}^n$, our objective is then to achieve a low-rank decomposition of these weights with the smallest possible ranks, formulated as the following optimization problem:

$$\min_{\widehat{\boldsymbol{\mathcal{W}}}^{\mathcal{R}}} \mathcal{L}_d(\widehat{\boldsymbol{\mathcal{W}}}^{\mathcal{R}}) \quad s.t. \quad \min_{\mathcal{R}} \mathcal{L}_r(\mathcal{R}), \tag{4}$$

where $\mathcal{L}_d(.)$ and $\mathcal{L}_r(.)$ are a decomposition loss and a rank loss, respectively, and $\widehat{\boldsymbol{\mathcal{W}}}^{\mathcal{R}} = \left\{\hat{\boldsymbol{\mathcal{W}}}_1^{(\mathcal{R}_1)}, \ldots, \hat{\boldsymbol{\mathcal{W}}}_n^{(\mathcal{R}_n)}\right\}$ is the set of decompositions to be found with $\mathcal{R} = \{\mathcal{R}_1, \ldots, \mathcal{R}_n\}$ the set of ranks, and $\hat{\boldsymbol{\mathcal{W}}}_i^{(\mathcal{R}_i)}$ are the weights of decomposition corresponding to the ranks $\mathcal{R}_i = \{r_i^1, ..., r_i^k\}$ of layer i.

For each layer $i \in \{1, \ldots, n\}$ of the network, we consider a set of decompositions $(\hat{\boldsymbol{\mathcal{W}}}_i^{(r)})_{r \in \mathcal{R}_i}$ of varying ranks defined in the set $\mathcal{R}_i$, for each weight tensor $\boldsymbol{\mathcal{W}}_i$. To make the optimization problem under the rank constraint (4) continuous, we associate a rank coefficient $p_i^{(r)}$ with each decomposition of rank r in layer i based on a learnable parameter $\alpha_i^{(r)}$.

This rank coefficient, defined as $p_i^{(r)} = \text{softmax}(\alpha_i^{(r)})$, is adjusted via the parameter $\alpha_i^{(r)}$ to reflect the probability that the rank r will be used in the decomposition of the weight tensor $\mathcal{W}_i$ for the layer i. Inspired by [31], we formalize the rank constraint in (4) using a normalized rank loss:

$$\mathcal{L}_r(\mathcal{R}) = \gamma \sum_{i=1}^{n} \left(\sum_{r \in \mathcal{R}_i} p_i^{(r)} \frac{r}{\max \mathcal{R}_i} \right)^{\beta}, \tag{5}$$

where $\beta, \gamma \in [0, 1]$ are hyperparameters.

4.2 Tensor Decomposition and Rank Exploration

Building on the definition of $\mathcal{L}_r(\mathcal{R})$, we introduce two total losses to update weights and parameters $(\alpha_i^{(r)})_{i,r}$. The total weights loss for a neural network model with n layers is defined as follows:

$$\mathcal{L}_{Tw}(\widehat{\boldsymbol{\mathcal{W}}}^{\mathcal{R}}, \mathcal{P}^{\mathcal{R}}) = \sum_{i=1}^{n} \left\| \boldsymbol{\mathcal{W}}_i - \sum_{r \in \mathcal{R}_i} p_i^{(r)} \hat{\boldsymbol{\mathcal{W}}}_i^{(r)} \right\|_F^2 \times \gamma \left[\sum_{i=1}^{n} \left(\sum_{r \in \mathcal{R}_i} p_i^{(r)} \frac{r}{\max \mathcal{R}_i} \right)^{\beta} \right], \tag{6}$$

and the total parameters loss for the same neural network can be formulated as:

$$\mathcal{L}_{T\alpha}(\widehat{\boldsymbol{\mathcal{W}}}^{\mathcal{R}}, \mathcal{P}^{\mathcal{R}}) = \mathcal{L}_{val}(\widehat{\boldsymbol{\mathcal{W}}}^{\mathcal{R}}, \mathcal{P}^{\mathcal{R}}) \times \gamma \left[\sum_{i=1}^{n} \left(\sum_{r \in \mathcal{R}_i} p_i^{(r)} \frac{r}{\max \mathcal{R}_i} \right)^{\beta} \right], \tag{7}$$

The first term in (6) is referred to as the *decomposition loss*, while $\mathcal{L}_{\text{val}}$ in (7) denotes the cross-entropy loss on the validation data. The minimization of the losses, is tackled through a two-step iterative process. First, the weights of the decomposition, denoted as $\widehat{\boldsymbol{\mathcal{W}}}^{\mathcal{R}}$, are updated by minimizing (6) while keeping the rank coefficients, denoted as $\mathcal{P}^{\mathcal{R}}$, fixed across all layers. Next, the parameters $(\alpha_i^{(r)})_{i,r}$, are updated using the newly updated weights $\widehat{\boldsymbol{\mathcal{W}}}^{\mathcal{R}}$. This update is performed by minimizing the (7), where the weights between the layers are adjusted with the corresponding rank parameters. This two-steps updates ensures that each α update is based on well-trained weights, avoiding noisy signals from still-learning weights. This prevents the architecture from overfitting to transient weight states.

The updates of the decomposition weight parameters and rank coefficients are performed using stochastic gradient descent to ensure efficient and iterative optimization. The update rules are as follows:

$$\text{Weight update:} \quad \hat{\boldsymbol{\mathcal{W}}}_i^{(r)} \leftarrow \hat{\boldsymbol{\mathcal{W}}}_i^{(r)} - \eta_w \nabla_{\hat{\boldsymbol{\mathcal{W}}}_i^{(r)}} (\mathcal{L}_{Tw}), \tag{8}$$

$$\text{Rank coefficient update:} \quad \alpha_i^{(r)} \leftarrow \alpha_i^{(r)} - \eta_\alpha \nabla_{\alpha_i^{(r)}} (\mathcal{L}_{T\alpha}). \tag{9}$$

For each layer i and each rank $r \in \mathcal{R}_i$, the weight update (8) and rank coefficient update (9) are performed iteratively until a local minimum of the total loss (6) is reached. Each loss is a multiplication combination, where decomposition and validation losses are scaled by the rank loss (and vice versa). This scale-invariant feature balances both terms without requiring separate trade-off hyperparameters, enabling more stable training and better results compared to additive combinations.

4.3 Rank Search Space

Previous approaches to rank search in neural network compression typically rely on evaluating a small, fixed set of candidate ranks. While this strategy offers computational efficiency, it risks overlooking the most optimal rank configurations, as the true optimum may lie between the preselected candidates. To address this limitation, we propose a multi-step rank search method that systematically explores the entire rank space and progressively refines the search around the most promising solutions.

The process begins by defining a broad search space for each layer, denoted as $\mathcal{R}_i$, which spans all feasible rank values from $r_{\min}$ to $r_{\max}$. An initial step size $s^{(0)}$ is chosen to sample candidate ranks at regular intervals across this range, ensuring a coarse but complete coverage of the search space. For each sampled rank, the network weights and associated rank coefficients are updated, allowing the model to adapt to the current rank configuration. The quality of each candidate rank is assessed according to a loss function that may include both reconstruction error and a regularization term to encourage lower ranks.

After this initial exploration, the method identifies, for each layer i, the rank $\bar{r}_i$ that achieves the highest rank coefficient, indicating its potential as a promising candidate. To focus the search more precisely, the algorithm then defines new lower and upper bounds, Lb_i and Ub_i, centered around $\bar{r}_i$ and separated by half the previous step size on either side. This effectively narrows the search space to a region most likely to contain the optimal rank. The step size is then reduced by a factor $f > 1$, yielding a finer sampling resolution for the next iteration. The new set of candidate ranks for layer i is thus given by:

$$\mathcal{R}_i = \{r \mid r = Lb_i + ks, \text{ for } k \in \mathbb{N}, \text{ and } Lb_i \leq r \leq Ub_i\},$$

where s denotes the updated step size. Before commencing the next iteration, the weights and rank coefficients are reinitialized for the refined search space. The process of sampling, updating, and selecting is then repeated. With each iteration, the search space contracts and the step size decreases, leading to an increasingly precise localization of the optimal rank. This iterative refinement continues until, for each layer, the candidate set $\mathcal{R}_i$ contains only a single element, signifying convergence to a unique rank selection.

Throughout this procedure, the weights and rank coefficients are jointly optimized, ensuring that both the model parameters and the rank configuration are adapted to minimize the overall loss. The loss function can incorporate not only the reconstruction or decomposition error but also a regularization component that penalizes higher ranks, thereby promoting model compression.

At the conclusion of the search, the final rank configuration $\bar{\mathbf{r}} = (\bar{r}_1, \ldots, \bar{r}_n)$ is validated using a cross-entropy loss or another appropriate metric on a held-out validation set. This step ensures that the selected ranks yield not only a compact model but also satisfactory predictive performance. The balance between compression and accuracy can be tuned by adjusting the regularization parameters γ and β in the loss formulation.

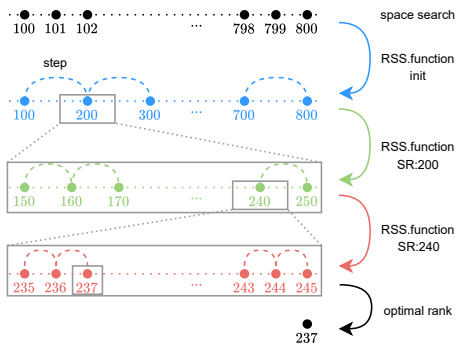

Fig. 3: A toy example illustrates the search for rank spaces. Initially, the search space includes integers from 100 to 800, with a step size of 100. After the first iteration, the selected rank is $\bar{r} = 200$, narrowing the search interval to $[150, 250]$ with a step size of 10. The second iteration selects $\bar{r} = 240$, refining the search space to $[235, 245]$ with a step size of 1. After 3 iterations, the optimal rank is identified within this interval.

This multi-step, progressive rank search method offers several advantages over traditional approaches. By systematically narrowing the search space and refining the sampling granularity, it combines the thoroughness of exhaustive search with the efficiency of adaptive optimization. The method is capable of escaping the limitations imposed by fixed candidate sets and can converge to globally optimal or near-optimal rank configurations. Figure 3 illustrates the evolution of the search space for a single layer: the process begins with a wide interval and large step size, then successively narrows and refines the search until the optimal rank is identified with maximal precision.

4.4 Final Decomposition and Fine-Tuning

The optimal ranks for decomposing the tensor weights for each layer, denoted as $\boldsymbol{\mathcal{R}}^\star = \{r_1^\star, \ldots, r_n^\star\}$, are determined from these final sets and used to construct the decomposed network. To ensure the decomposed model replicates the behavior of the original model, it is crucial that the layers not only align their decomposed weights with the original weights but also produce the same outputs. To achieve this, our fine-tuning loss ($\mathcal{L}_f$) consists of two components: a cross entropy loss ($\mathcal{L}_{ce}$) and distillation loss ($\mathcal{L}_{diss}$). The cross entropy loss adjusts the model's weights based on the training data labels. Distillation loss aligns the decomposed weights with the original weights by minimized Frobenius distance, and enforces consistency between the outputs of the original and decomposed layers. The distillation and fine-tuning losses are defined as follows:

$$\mathcal{L}_{diss}(\widehat{\boldsymbol{\mathcal{W}}}^{\boldsymbol{\mathcal{R}}^\star}) = \sum_{i=1}^{n} \left\| \boldsymbol{\mathcal{W}}_i - \hat{\boldsymbol{\mathcal{W}}}_i^{(r_i^\star)} \right\|_F^2 + \sum_{x \in X} \sum_{i=1}^{n} \|O_i(x) - D_i(x)\|_F^2 , \tag{10}$$

Algorithm 1: Rank adapt tENsor dEcomposition (`RENE`)

Input: Pretrained model M, Training data X, Rank lower bounds $\boldsymbol{Lb} = \{Lb_1, \ldots, Lb_n\}$ and upper bounds $\boldsymbol{Ub} = \{Ub_1, \ldots, Ub_n\}$, Number of iterations T, Step size $s > 1$, Factor f;
Initialize: $\forall i, \mathcal{R}_i \leftarrow \{r \mid r = Lb_i + ks, \text{ for } k \in \mathbb{N}, \text{ and } Lb_i \leq r \leq Ub_i\}$;
while $s > 1$ **do**
  for $i \in \{1, \ldots, n\}$ **do**
    for $r \in \mathcal{R}_i$ **do**
      for $t = 1$ ***to*** T **do**
        $\hat{\boldsymbol{\mathcal{W}}}_i^{(r)} \leftarrow update(\hat{\boldsymbol{\mathcal{W}}}_i^{(r)})$; `// Eq.` (8)
        $\alpha_i^{(r)} \leftarrow update(\alpha_i^{(r)})$; `// Eq.` (9)
  $s \leftarrow \left\lfloor \frac{s}{f} \right\rfloor$;
  for $i \in \{1, \ldots, n\}$ **do**
    $\bar{r}_i \leftarrow argmax_{r \in \mathcal{R}_i}(\text{softmax}(\alpha_i^{(r)}))$;
    $Lb_i \leftarrow \bar{r}_i - \frac{s}{2}$;
    $Ub_i \leftarrow \bar{r}_i + \frac{s}{2}$;
    $\mathcal{R}_i \leftarrow \{r \mid r = Lb_i + ks, \text{ for } k \in \mathbb{N}, \text{ and } Lb_i \leq r \leq Ub_i\}$;
Output: Decomposed model M^* by minimizing $\mathcal{L}_f(\widehat{\boldsymbol{\mathcal{W}}}^{\boldsymbol{\mathcal{R}^\star}})$ using X ; `// Eq.` (11)

$$\mathcal{L}_f(\widehat{\boldsymbol{\mathcal{W}}}^{\boldsymbol{\mathcal{R}^\star}}) = \mathcal{L}_{ce} + \lambda \mathcal{L}_{diss}, \tag{11}$$

where X is the training set, $O_i(.)$ and $D_i(.)$ are the outputs of layer i of the original model and the decomposed one, respectively and λ is hyperparameter to control combination of losses. In this approach, the original model serves as the teacher model and the decomposed model acts as the student model. The pseudocode for the overall procedure retracing these steps is presented in Algorithm 1.

5 Experiments

5.1 Experimental Setup

We evaluate `RENE`[1] on 3 datasets including CIFAR-10/100 [17] and ImageNet-1K [9]. To prevent convergence collapse during the updating of Eq.(6) and Eq.(7), we initially update only the weights for several iterations before jointly updating both weights and rank coefficients in an iterative manner. Each experiment is performed five times, and the best result from the fine-tuning step is reported. For TT decomposition, due to computational resource constraints, we assume that the two TT ranks are equal. In the search phase of `RENE`, for CIFAR-10/100,

[1] The code is available for research purposes at https://github.com/aah94/RENE

we set the initial rank space to $\{10, \ldots, 100\}$ with a step size of $s = 10$, which corresponds to 2 search steps. For ImageNet-1K, we set the initial rank space to $\{50, \ldots, 850\}$ with the step size $s = 100$, corresponding to 3 search steps. Across all datasets, we use $f = 10$. We used the standard SGD optimizer with Nesterov momentum set to 0.9, and hyperparameters λ, γ and β set to 0.5, 0.4 and 0.8, respectively. The initial learning rates were 0.001 for CIFAR-10/100 and 0.0001 for ImageNet-1K. For the fine-tuning step we consider learning rate 0.00001 for all experiments and grid search with cross-validation is employed to select all hyperparameters, optimizing model performance based on validation accuracy. For comparing different approaches, the TOP-1 accuracy is used to compare the performance of the compressed model against the original uncompressed model. Additionally, we consider the gain in floating operations per second (FLOPs) and the compression rate.

5.2 Experimental Results

The following sections present a comprehensive analysis of `RENE`'s performance and compression capabilities across various models and datasets.

Performance and Compression Analysis. For the initial evaluation, we tested `RENE` on CIFAR-10 using the ResNet-20 and VGG-16 models, with the results presented in Table 1. `RENE` with both CP and TT decomposition techniques yields competitive results compared to state-of-the-art methods. Using ResNet-20 as the original model, `RENE` with CP decomposition achieves 1.24% and 1.52% greater reduction of FLOPs and parameters, respectively, compared to the HALOC method [33]. Additionally, `RENE` with TT decomposition improves accuracy by 0.08% over the original uncompressed model. This suggests that our approach has effectively reduced the number of parameters of the original model, leading to a better generalization. Furthermore, with the VGG-16 model, `RENE` achieves significant compression rates while preserving performance. For instance, using `RENE` with CP decomposition reduces FLOPs by 85.23% and parameters by 98.6%. Furthermore, applying `RENE` with TT decomposition on VGG-16 improves generalization, resulting in a 0.04% increase in TOP-1 accuracy compared to the original uncompressed model.

The results on the ImageNet-1K dataset are presented in Table 2, where we evaluated `RENE` using ResNet-18 and MobileNetV2 models. For ResNet-18, our approach with CP decomposition yields competitive results, while TT decomposition outperformed other methods, achieving state-of-the-art performance across all metrics, including Top-1 accuracy, reduction in FLOPs, and parameters. With MobileNetV2, the CP method did not yield high performance, likely due to the model's reliance on depthwise convolution, which does not significantly benefit from decomposition in certain dimensions. However, `RENE` with TT decomposition demonstrated superior compression results, achieving 1.86% and 2.31% greater reductions in FLOPs and parameters, respectively, along with competitive Top-1 accuracy. Our results underscore the importance of selecting

Table 1: Results of different compression approaches for ResNet-20 and VGG-16 on CIFAR-10. C.T and A.R stand for *compression technique* and *automatic rank*, respectively.

Method	C.T	A.R	Top-1	FLOPs (↓%)	Comp. Rate
ResNet-20	Original	-	91.25	-	-
RENE(CP)	Low-rank	✓	90.82	**73.44**	**77.62**
RENE(TT)	Low-rank	✓	**91.40**	70.4	72.28
HALOC [33]	Low-rank	✓	91.32	72.20	76.10
ALDS [21]	Low-rank	✓	90.92	67.86	74.91
LCNN [15]	Low-rank	✓	90.13	66.78	65.38
PSTR-S [20]	Low-rank	✓	90.80	65.00	60.87
Std. Tucker [16]	Low-rank	✗	87.41	62.00	61.54
VGG-16	Original	-	92.78	-	-
RENE(CP)	Low-rank	✓	92.51	86.23	**98.60**
RENE(TT)	Low-rank	✓	**93.20**	86.10	95.51
HALOC [33]	Low-rank	✓	93.16	**86.44**	98.56
ALDS [21]	Low-rank	✓	92.67	86.23	95.77
LCNN [15]	Low-rank	✓	92.72	85.47	91.14
DECORE [1]	Pruning	-	92.44	81.50	96.60
Spike-Thrift [18]	Pruning	-	91.79	80.00	97.01

Table 2: Results of different compression approaches for ResNet-18 and MobileNetV2 on ImageNet-1K.

Method	C.T	A.R	Top-1	FLOPs (↓%)	Comp. Rate
ResNet-18	Original	-	69.75	-	-
RENE(CP)	Low-rank	✓	68.46	57.1	66.2
RENE(TT)	Low-rank	✓	**70.88**	**68.9**	**67.1**
HALOC [33]	Low-rank	✓	70.65	66.16	63.64
ALDS [21]	Low-rank	✓	69.22	43.51	66.70
TETD [37]	Low-rank	✗	69.00	59.51	60.00
Stable EPC [25]	Low-rank	✓	68.50	59.51	61.00
MUSCO [12]	Low-rank	✗	69.29	58.67	60.50
CHEX [14]	Pruning	-	69.60	43.38	59.00
EE [39]	Pruning	-	68.27	46.60	58.00
SCOP [28]	Pruning	-	69.18	38.80	39.30
MobileNetV2	Original	-	71.85	-	-
RENE(CP)	Low-rank	✓	65.39	11.78	**51.6**
RENE(TT)	Low-rank	✓	70.1	**26.7**	42.34
HALOC [33]	Low-rank	✓	**70.98**	24.84	40.03
ALDS [21]	Low-rank	✓	70.32	11.01	32.97
HOSA [28]	Pruning	-	64.43	43.65	91.14
DCP [7]	Pruning	-	64.22	44.75	96.60
FT [40]	Pruning	-	70.12	20.23	21.31

the appropriate decomposition method based on the model's complexity. Our experiments indicate that TT decomposition is more effective for compressing higher-complexity models, such as those trained on the ImageNet-1K dataset, while CP decomposition excels in compressing lower-complexity models, like those classically used on CIFAR-10.

Automatic vs. Manual Rank Selection. We now examine the effectiveness of our rank search process compared to manual rank setting. In this experiment, we used pretrained ResNet18 and VGG16 models on the CIFAR-10 and CIFAR-100 datasets. For manual rank setting, we apply the TT decomposition and fix the rank across all layers to achieve a decomposed model with a specific percent-

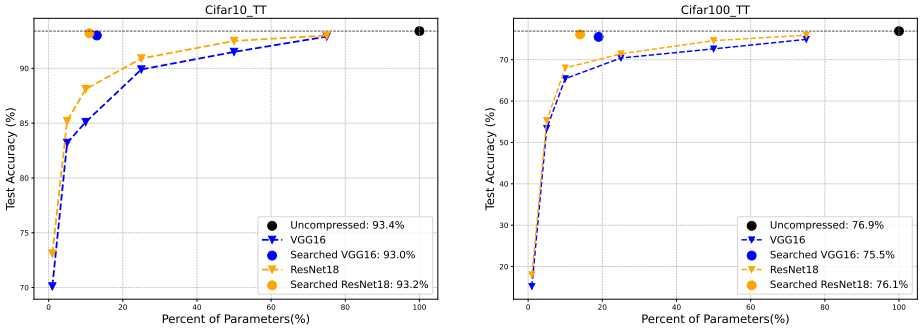

Fig. 4: Search vs Manual: Compression results for manual setting at different levels compression, compare to searched setting (left CIFAR10 and right CIFAR100).

age of the initial model's parameters, chosen from the set $\{1, 5, 10, 25, 50, 75\}$. All models are pre-trained on the ImageNet-1K dataset and fine-tuned for 20 epochs. Figure 4 presents these results. As shown, increasing the number of ranks (or equivalently, increasing the percentage of parameters of the decomposed model) improves the performance of both the decomposed VGG16 and ResNet18 models. When the decomposed models have 75% of the parameters of the initial models, the performance almost matches that of the original pretrained models. With `RENE`, We achieve comparable results while compressing the model by more than 80% on both ResNet18 and VGG16 across both datasets. These results indicate that fixing the ranks across layers is suboptimal. In contrast, `RENE` enables the automatic selection of ranks across different layers, achieving a good compression rate without significant performance loss.

Rank selection Figure 5 illustrates the selected ranks for both CP and TT decompositions using ResNet-18 as the original model on the ImageNet-1K dataset, highlighting that CP ranks are generally larger than those of TT.

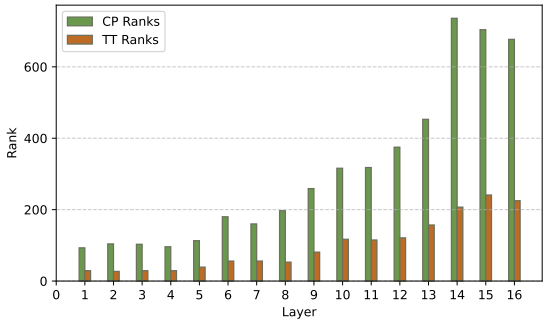

Fig. 5: Distribution of ranks achieved using CP and TT decompositions on ResNet-18 for the ImageNet-1K dataset.

This difference arises from the inherent characteristics of the decomposition methods: CP decomposition tends to produce larger ranks because it decomposes the tensor into a sum of rank-one tensors, capturing more detailed interactions but potentially leading to higher complexity. In contrast, TT decomposition typically results in smaller ranks due to its chain-like structure, which can lead to more compact representations and potentially better compression. The distribution of ranks reveals that even among layers of the same dimensions, the effective ranks can differ. This reflects the varying contributions of each layer to the model's performance. Some layers may capture more complex features, requiring higher ranks, while others may focus on simpler features, allowing for lower ranks. These results are in line with the case of selecting the ranks manually and the same over all layers that were been presented in the previous section.

Double Compression In this experiment, we investigate the effects of double compression by applying `RENE` in conjunction with knowledge distillation. Our goal is to assess whether combining these two compression techniques can yield further reductions in model size and computational requirements without sacrificing performance. We focus on TT decomposition for this analysis, using two datasets: CIFAR-100 and ImageNet-1K.

For the CIFAR-100 dataset, we employ ResNet-56 as the teacher model and ResNet-20 as the student model. Similarly, for the ImageNet-1K dataset, ResNet-34 serves as the teacher model, while ResNet-18 acts as the student model. The distillation process involves training the student model to mimic the behavior of the larger, more complex teacher model, thereby transferring knowledge and improving performance.

Table 3: Double compression: `RENE` with distillation on CIFAR-100 and ImageNet-1K. The notations T and S denote the teacher and student, respectively.

CIFAR-100 (T: ResNet56 (72.34%), S: ResNet20 (69.6%))			
Method	**Top-1 (%)**	**FLOPs (%)**	**Comp. rate (%)**
Distillation [30]	**72.53**	67.7	68.24
`RENE`(Teacher)	72.23	64.23	61.75
`RENE`(Student)	72.46	**89.01**	**86.54**
ImageNet-1K (T: ResNet34 (73.31%), S: ResNet18 (69.76%))			
Method	**Top-1 (%)**	**FLOPs (%)**	**Params (%)**
Distillation [30]	71.98	50.27	46.33
`RENE`(Teacher)	**73.23**	59.91	63.46
`RENE`(Student)	71.9	**76.77**	**78.69**

After applying distillation, we further compress both the teacher and the distilled student models using `RENE`. The results, presented in Table 3, demonstrate that our decomposition method achieves competitive performance compared to distillation alone for both the teacher and student models. Notably, when applying `RENE` to the distilled student model, we achieve a significant reduction in both parameters and computational complexity. Specifically, on the ImageNet-1K dataset, the decomposed distilled student model reduces parameters by 78.69% and FLOPs by 76.77% compared to the original teacher model.

This double compression approach not only maintains the accuracy of the original model but also highlights the potential for substantial reductions in model size and computational requirements. These findings underscore the effectiveness of combining distillation with decomposition techniques to achieve efficient and high-performing compressed models.

6 Conclusion

In this paper, we presented an approach for compressing deep neural networks through decomposition and optimal rank selection. Our solution stands out with two key features: it considers all layers during the optimization process, aiming for high compression rates without compromising accuracy by identifying the optimal rank pattern across layers. This approach capitalizes on the varying contributions of different layers to the model's inference, allowing for smaller ranks in less critical layers and determining the most effective rank pattern for each. To achieve significant compression, we explore a broad range of ranks, addressing the substantial memory challenges of this extensive exploration with a multistage rank search strategy. This strategy enables comprehensive exploration while ensuring efficient memory usage. Our experimental results demonstrate that this approach effectively reduces the number of parameters and computational complexity, leading to better generalization and competitive performance across various models and datasets.

References

1. Alwani, M., Wang, Y., Madhavan, V.: Decore: Deep compression with reinforcement learning. In: Proceedings of the Conference on Computer Vision and Pattern Recognition. pp. 12349–12359 (2022)
2. Audibert, A., Amini, M.R., Usevich, K., Clausel, M.: Low-rank updates of pretrained weights for multi-task learning. In: Findings of the Association for Computational Linguistics: ACL. pp. 7544–7554 (2023)
3. Beyer, L., Zhai, X., Royer, A., Markeeva, L., Anil, R., Kolesnikov, A.: Knowledge distillation: A good teacher is patient and consistent. In: Proceedings of the conference on computer vision and pattern recognition. pp. 10925–10934 (2022)
4. Blalock, D., Gonzalez Ortiz, J.J., Frankle, J., Guttag, J.: What is the state of neural network pruning? Proceedings of machine learning and systems **2**, 129–146 (2020)
5. Cao, T., Sun, L., Nguyen, C.H., Mamitsuka, H.: Learning low-rank tensor cores with probabilistic ℓ_0-regularized rank selection for model compression. In: Proceedings of the 33^{rd} International Joint Conference on Artificial Intelligence, IJCAI. pp. 3780–3788 (2024)
6. Chang, C.C., Sung, Y.Y., Yu, S., Huang, N.C., Marculescu, D., Wu, K.C.: Flora: Fine-grained low-rank architecture search for vision transformer. In: Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision. pp. 2482–2491 (2024)
7. Chatzikonstantinou, C., Papadopoulos, G.T., Dimitropoulos, K., Daras, P.: Neural network compression using higher-order statistics and auxiliary reconstruction losses. In: Proceedings of the Conference on Computer Vision and Pattern Recognition Workshops. pp. 716–717 (2020)
8. Dai, W., Fan, J., Miao, Y., Hwang, K.: Deep learning model compression with rank reduction in tensor decomposition. IEEE Transactions on Neural Networks and Learning Systems (2023)
9. Deng, J., Russakovsky, O., Krause, J., Bernstein, M., Berg, A.C., Fei-Fei, L.: Scalable multi-label annotation. In: ACM Conference on Human Factors in Computing Systems (CHI) (2014)

10. Domanov, I., De Lathauwer, L.: Canonical polyadic decomposition of third-order tensors: Relaxed uniqueness conditions and algebraic algorithm. Linear Algebra and its Applications **513**, 342–375 (2017)
11. Goldfarb, D., Qin, Z.: Robust low-rank tensor recovery: Models and algorithms. SIAM Journal on Matrix Analysis and Applications **35**(1), 225–253 (2014)
12. Gusak, J., Kholiavchenko, M., Ponomarev, E., Markeeva, L., Blagoveschensky, P., Cichocki, A., Oseledets, I.: Automated multi-stage compression of neural networks. In: Proceedings of the IEEE/CVF International Conference on Computer Vision Workshops. pp. 0–0 (2019)
13. Hillar, C.J., Lim, L.H.: Most tensor problems are np-hard. Journal of the ACM (JACM) pp. 1–39 (2013)
14. Hou, Z., Qin, M., Sun, F., Ma, X., Yuan, K., Xu, Y., Chen, Y.K., Jin, R., Xie, Y., Kung, S.Y.: Chex: Channel exploration for cnn model compression. In: Proceedings of the Conference on Computer Vision and Pattern Recognition. pp. 12287–12298 (2022)
15. Idelbayev, Y., Carreira-Perpinán, M.A.: 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)
16. Kim, Y.D., Park, E., Yoo, S., Choi, T., Yang, L., Shin, D.: Compression of deep convolutional neural networks for fast and low power mobile applications. In: 4th International Conference on Learning Representations, ICLR (2016)
17. Krizhevsky, A., Hinton, G., et al.: Learning multiple layers of features from tiny images (2009)
18. Kundu, S., Datta, G., Pedram, M., Beerel, P.A.: Spike-thrift: Towards energy-efficient deep spiking neural networks by limiting spiking activity via attention-guided compression. In: Proceedings of the IEEE/CVF winter conference on applications of computer vision. pp. 3953–3962 (2021)
19. Li, H., Kadav, A., Durdanovic, I., Samet, H., Graf, H.P.: Pruning filters for efficient convnets. In: International Conference on Learning Representations, ICLR (2017)
20. Li, N., Pan, Y., Chen, Y., Ding, Z., Zhao, D., Xu, Z.: Heuristic rank selection with progressively searching tensor ring network. Complex & Intelligent Systems pp. 1–15 (2021)
21. Liebenwein, L., Maalouf, A., Feldman, D., Rus, D.: Compressing neural networks: Towards determining the optimal layer-wise decomposition. Advances in Neural Information Processing Systems (2021)
22. Novikov, A., Podoprikhin, D., Osokin, A., Vetrov, D.P.: Tensorizing neural networks. Advances in neural information processing systems **28** (2015)
23. Oseledets, I.V.: Tensor-train decomposition. SIAM Journal on Scientific Computing **33**(5), 2295–2317 (2011)
24. Park, E., Ahn, J., Yoo, S.: Weighted-entropy-based quantization for deep neural networks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 5456–5464 (2017)
25. Phan, A.H., Sobolev, K., Sozykin, K., Ermilov, D., Gusak, J., Tichavskỳ, P., Glukhov, V., Oseledets, I., Cichocki, A.: Stable low-rank tensor decomposition for compression of convolutional neural network. In: Computer Vision–ECCV. pp. 522–539 (2020)
26. Radford, A., Kim, J.W., Hallacy, C., Ramesh, A., Goh, G., Agarwal, S., Sastry, G., Askell, A., Mishkin, P., Clark, J., et al.: Learning transferable visual models from natural language supervision. In: International conference on machine learning (2021)

27. Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., Chen, L.C.: Mobilenetv2: Inverted residuals and linear bottlenecks. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 4510–4520 (2018)
28. Tang, Y., Wang, Y., Xu, Y., Tao, D., Xu, C., Xu, C., Xu, C.: Scop: Scientific control for reliable neural network pruning. Advances in Neural Information Processing Systems **33**, 10936–10947 (2020)
29. Tucker, L.R.: Some mathematical notes on three-mode factor analysis. Psychometrika **31**(3), 279–311 (1966)
30. Wang, Y., Cheng, L., Duan, M., Wang, Y., Feng, Z., Kong, S.: Improving knowledge distillation via regularizing feature norm and direction. In: Computer Vision - ECCV. pp. 20–37 (2024)
31. Wu, B., Dai, X., Zhang, P., Wang, Y., Sun, F., Wu, Y., Tian, Y., Vajda, P., Jia, Y., Keutzer, K.: Fbnet: Hardware-aware efficient convnet design via differentiable neural architecture search. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. pp. 10734–10742 (2019)
32. Xiao, J., Yin, M., Gong, Y., Zang, X., Ren, J., Yuan, B.: Comcat: Towards efficient compression and customization of attention-based vision models. In: ICML. pp. 38125–38136 (2023), https://proceedings.mlr.press/v202/xiao23e.html
33. Xiao, J., Zhang, C., Gong, Y., Yin, M., Sui, Y., Xiang, L., Tao, D., Yuan, B.: Haloc: hardware-aware automatic low-rank compression for compact neural networks. In: Proceedings of the AAAI Conference on Artificial Intelligence. pp. 10464–10472 (2023)
34. Xu, L., Cheng, L., Wong, N., Wu, Y.C.: Tensor train factorization under noisy and incomplete data with automatic rank estimation. Pattern Recognition **141**, 109650 (2023)
35. Yang, Y., Krompass, D., Tresp, V.: Tensor-train recurrent neural networks for video classification. In: International Conference on Machine Learning. pp. 3891–3900. PMLR (2017)
36. Yin, M., Phan, H., Zang, X., Liao, S., Yuan, B.: Batude: Budget-aware neural network compression based on tucker decomposition. In: AAAI Conference on Artificial Intelligence. pp. 8874–8882 (2022)
37. Yin, M., Sui, Y., Liao, S., Yuan, B.: Towards efficient tensor decomposition-based dnn model compression with optimization framework. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 10674–10683 (2021)
38. Yu, X., Liu, T., Wang, X., Tao, D.: 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)
39. Zhang, Y., Gao, S., Huang, H.: Exploration and estimation for model compression. In: Proceedings of the International Conference on Computer Vision. pp. 487–496 (2021)
40. Zhuang, Z., Tan, M., Zhuang, B., Liu, J., Guo, Y., Wu, Q., Huang, J., Zhu, J.: Discrimination-aware channel pruning for deep neural networks. Advances in neural information processing systems **31** (2018)