Cadre unifié pour la compression de réseaux neuronaux pré-entraînés via la décomposition et la sélection de rang optimisée

Ali Aghababaei-Harandi, Massih-Reza Amini

Cadre unifié pour la compression de réseaux neuronaux pré-entraînés via la décomposition et la sélection de rang optimisée

Ali AGHABABAEI-HARANDI, Massih-Reza AMINI
Université Grenoble Alpes, CNRS, Computer Science Laboratory LIG, Grenoble, France
Firstname.LASTNAME@univ-grenoble-alpes.fr

Abstract

Malgré leur grande précision, les réseaux de neurones de grande taille nécessitent des ressources de calcul considérables, ce qui rend leur déploiement difficile sur des appareils aux capacités limitées, comme les téléphones portables ou les systèmes embarqués. Pour relever ce défi, des algorithmes de compression ont été développés afin de réduire la taille des modèles et leurs besoins en calcul, tout en préservant leur précision. Parmi ces approches, les méthodes de factorisation basées sur la décomposition tensorielle se distinguent par leur solidité théorique et leur efficacité. Cependant, elles peinent à déterminer le rang optimal pour la décomposition. Cet article propose un cadre unifié qui combine simultanément la décomposition et la sélection du rang optimal, en utilisant une fonction de perte de compression composite sous des contraintes de rang définies. Notre méthode intègre une recherche automatique du rang dans un espace continu, permettant d'identifier efficacement les configurations optimales sans recourir à des données d'apprentissage. Cette approche est ainsi économiquement avantageuse en termes de calcul. Associée à une étape ultérieure de réglage fin, elle maintient les performances des modèles fortement compressés à un niveau comparable à celui des modèles originaux. Les résultats expérimentaux, obtenus sur divers jeux de données de référence, démontrent l'efficacité de notre méthode par rapport aux techniques de l'état de l'art.

Keywords

Réseaux de neurones, Décomposition, Rang optimal.

1 Introduction

In recent years, deep learning has revolutionized various scientific fields, including computer vision and natural language processing [30, 11]. 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 [31]. 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 [28]. 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 [5, 35]. *Quantization* reduces the precision of numerical values, typically transitioning from 32-bit floating-point numbers to lower bit-width fixed-point numbers [38, 27]. *Knowledge distillation* trains a smaller "student" model to mimic a larger "teacher" model, resulting in a compact model with similar performance [4]. Lastly, *low-rank factorization* decomposes weight matrices or tensors into smaller components, reducing the number of parameters [41, 42]. 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 [15], and allow different decompositions of a same tensor even though some works tries to approximate ranks of tensor in practical manner[40, 13, 34],. 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 [6, 23, 39]. 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, which makes it computationally efficient.
— 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) [17]. These methods provide an efficient means of fine-tuning large language models, offering advantages over alternative techniques such as quantization [38], knowledge distillation [4], and gradient-based pruning [43]. Beyond compression itself, the way latent structures such as topics or segments are discovered and represented in models is critical. For example, [2] demonstrated that segmentation strategies applied to topic models improve topical coherence, which parallels the need for interpretable and efficient representations in low-rank model compression. Similarly, work by [3] on sentiment classification underscores the importance of effective representation learning in NLP tasks, motivating the use of advanced tensor decomposition methods to balance compression and performance.

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 [42]. However, selecting the appropriate rank for compressing deep neural models using decomposition techniques is NP-hard [15]. Research in this area falls into two main approaches.

The first approaches rely 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 [44]. The two main low-rank approximation methods applied on pre-trained models are CP and Tucker decomposition [19]. 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 [29]. In addition, some work decomposed convolution and fully connected layers with tensor train, and trained the model from scratch [25]. 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 [22], which can result in suboptimal approximations that can lead to accuracy drops or insufficient compression rates.

The second approaches involve 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 [14]. The discrete nature of rank search lends itself to discrete search algorithms, such as reinforcement learning and progressive search, to identify optimal ranks [23]. Other methods impose constraints on ranks and budget, using iterative optimization strategies [43]. More recent studies explore continuous search spaces to determine optimal ranks [9, 39, 7].

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 multi-dimensional 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) [12], Tucker [33], tensor train (TT) [26], and tensor ring (TR) decomposition [46].

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 [25]. 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.

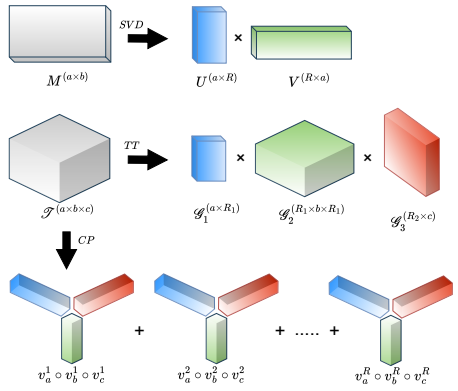

FIGURE 1 – An illustration of matrix decomposition (**upper row**) using SVD for a matrix $M \in \mathbb{R}^{a\times b}$, alongside Tensor Train decomposition (**middle row**), and CP decomposition (**bottom row**) for a tensor $\mathcal{T} \in \mathbb{R}^{a\times b\times c}$.

This decomposition mathematically can be represented as follows :

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

where the tuple $(R_1, R_2, \dots, R_{N-1})$ 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 $\mathcal{W} \in R^{b\times h\times w\times c}$, the forward process for an input tensor $\mathcal{X} \in R^{k_1\times k_2\times k_3}$ can be expressed as :

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

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 :

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

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 [12]. 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 $i \in \{1,\dots,n\}$, 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.

4.1 Problem Formulation

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

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

where $\mathcal{L}_d(.)$ and $\mathcal{L}_r(.)$ are a decomposition loss $\mathcal{L}_d$ and a rank loss, respectively, and $\widehat{\mathcal{W}}^{\mathcal{R}} = \left\{\hat{\mathcal{W}}_1^{(\mathcal{R}_1)},\dots,\hat{\mathcal{W}}_n^{(\mathcal{R}_n)}\right\}$ is the set of decompositions to be found with $\mathcal{R} = \{\mathcal{R}_1,\dots,\mathcal{R}_n\}$ the set of ranks, and $\hat{\mathcal{W}}_i^{(\mathcal{R}_i)}$ are the weights of decomposition corresponding to the ranks $\mathcal{R}_i = \{r_i^1,...,r_i^k\}$ of layer i. For each layer $i \in \{1,\dots,n\}$ of the network, we consider a set of decompositions $(\hat{\mathcal{W}}_i^{(r)})_{r\in\mathcal{R}_i}$ of varying ranks defined in the set $\mathcal{R}_i$, for each weight tensor $\mathcal{W}_i$.

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 likelihood that rank r will be used in the decomposition of the weight tensor $\mathcal{W}_i$ for layer i. Inspired by [37], we formalize the rank constraint in (4) using a normalized rank loss :

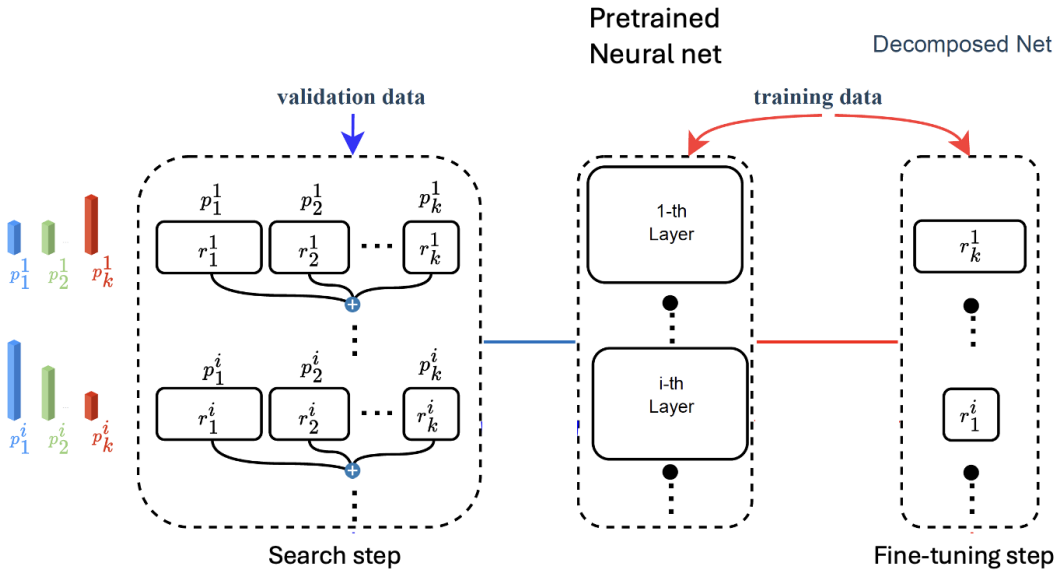

FIGURE 2 – Overview of RENE achieved in two phases : the rank search step (**left**), and the fine-tuning step (**right**).

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

where $\beta \in [0, 1]$ is a hyperparameter. Building on the definition of $\mathcal{L}_r(\mathcal{R})$, we introduce the total loss for a neural network model with n layers as follows :

$$\mathcal{L}_{Tw}(\widehat{\mathcal{W}}^{\mathcal{R}}, \mathcal{R}) = \sum_{i=1}^{n} \left\| \mathcal{W}_i - \sum_{r \in \mathcal{R}_i} p_i^{(r)} \hat{\mathcal{W}}_i^{(r)} \right\|_F^2 * \gamma \mathcal{L}_r(\mathcal{R}) \tag{6}$$

where $\gamma \in (0, 1]$ is a hyperparameter that balances the trade-off between decomposition loss and rank loss.

4.2 Tensor Decomposition and Rank Exploration

The minimization of the total loss function, as defined in equation (6), is tackled through a two-step iterative process. First, the weights of the decomposition, denoted as $\widehat{\mathcal{W}}^{\mathcal{R}}$, are updated 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 $\widehat{\mathcal{W}}^{\mathcal{R}}$. This update is performed by minimizing the total validation loss of all decomposed models, where the weights between the layers are adjusted with the corresponding rank parameters :

$$\mathcal{L}_{T^\alpha}(\widehat{\mathcal{W}}^{\mathcal{R}}, \mathcal{P}^{\mathcal{R}}) = \mathcal{L}_{val}(\widehat{\mathcal{W}}^{\mathcal{R}}, \mathcal{P}^{\mathcal{R}}) * \gamma \mathcal{L}_r(\mathcal{R}). \tag{7}$$

Where $\mathcal{L}_{val}(.)$ is loss cross entropy on validation data. 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 :

$$\mathcal{W} \text{ update :} \quad \hat{\mathcal{W}}_{i+1}^{(r)} \leftarrow \hat{\mathcal{W}}_i^{(r)} - \eta_w \nabla_{\hat{\mathcal{W}}_i^{(r)}} (\mathcal{L}_{Tw}), \tag{8}$$

$$\alpha \text{ update :} \quad \alpha_{i+1}^{(r)} \leftarrow \alpha_i^{(r)} - \eta_{\alpha_i^{(r)}} \nabla_{\alpha_i^{(r)}} (\mathcal{L}_{T^\alpha}). \tag{9}$$

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.

4.3 Rank Search Space

After updating the weight and rank coefficient, a rank $\bar{r}_i$ is selected for each layer i based on the highest rank coefficient obtained. The lower and upper bounds, Lb_i and Ub_i, are then set at a distance of half the step size around the selected rank. The size of the step s is set by dividing the size of the previous step by 10 and the new set of ranks for layer i is defined as the integers between Lb_i and Ub_i, spaced by s :

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

This iterative process of updating weights and rank coefficients, along with refining the rank search space, continues until the set of ranks $\mathcal{R}_i$ contains only one element.
Figure 3 illustrates this process for a single layer. The initial search space spans a broad interval with a large step size. After the first iteration, the search narrows around the selected rank, reducing the step size by a factor of 10. In the next iterations, the search space is further refined to a smaller, more precise range, ultimately reaching a step size of one for the final selection.

4.4 Final Decomposition and Fine-Tuning

The optimal ranks for decomposing the tensor weights for each layer, denoted as $\mathcal{R}^\star = \{r_1^\star, \ldots, r_n^\star\}$, are determined from these final sets and used to construct the decomposed network. To ensure the decomposed model replicates the

Algorithme 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 $s > 1$;

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)

  $s \leftarrow \lfloor \frac{s}{10} \rfloor$;

  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)

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 ($\mathcal{L}_D$). 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 forubinus distance, and enforces consistency between the outputs of the original and decomposed layers. The distillation and fine-tuning losses are defined as follows :

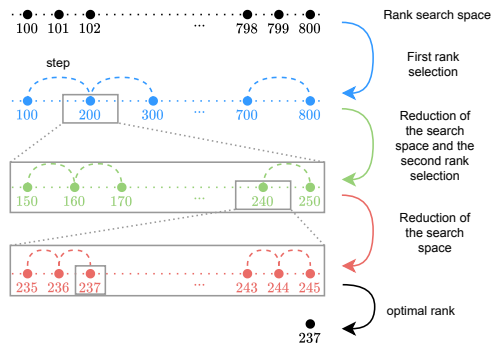

FIGURE 3 – A toy example illustrating the search of rank spaces. The initial search space includes all integers between 100 and 800. The first search for ranks begins with a step size of 100. After the first iteration, the selected rank is $\bar{r} = 200$, and the new interval is $[150, 250]$ with a step size of 10. Following the second iteration, the selected rank is $\bar{r} = 240$, and the final search space is $[235, 245]$ with a step size of 1. After 3 iterations, the optimal rank is identified within this interval.

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

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

where X is training dataset, $O_i(.)$ and $D_i(.)$ are the outputs of layer i of the original model and the decomposed one, respectively, 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

In the following section, we outline the methodology and configurations used in our experiments. We detail the datasets, models, and hyperparameters employed to evaluate the performance and efficiency of our proposed approach. We then present the outcomes of our experiments, highlighting the effectiveness of our approach compared to baseline methods. We analyze the results in terms of accuracy, computational efficiency, and compression rates, demonstrating the advantages of our adaptive rank selection process

5.1 Experimental Setup

We evaluate RENE on 3 datasets including CIFAR-10/100 [20] and ImageNet-1K [10] considering convolution and transformer-based models. 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, 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 same step size, corresponding to 3 search steps. 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. For comparing different approaches, the TOP-1 accuracy is used to compare the performance of the compressed model against

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 [39]	Low-rank	✓	91.32	72.20	76.10
ALDS [24]	Low-rank	✓	90.92	67.86	74.91
LCNN [18]	Low-rank	✓	90.13	66.78	65.38
PSTR-S [23]	Low-rank	✓	90.80	65.00	60.87
Std. Tucker [19]	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 [39]	Low-rank	✓	93.16	86.44	98.56
ALDS [24]	Low-rank	✓	92.67	86.23	95.77
LCNN [18]	Low-rank	✓	92.72	85.47	91.14
DECORE [1]	Pruning	-	92.44	81.50	96.60
Spike-Thrift [21]	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 [39]	Low-rank	✓	70.65	66.16	63.64
ALDS [24]	Low-rank	✓	69.22	43.51	66.70
TETD [43]	Low-rank	✗	69.00	59.51	60.00
Stable EPC [29]	Low-rank	✓	68.50	59.51	61.00
MUSCO [14]	Low-rank	✗	69.29	58.67	60.50
CHEX [16]	Pruning	-	69.60	43.38	59.00
EE [45]	Pruning	-	68.27	46.60	58.00
SCOP [32]	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 [39]	Low-rank	✓	70.98	24.84	40.03
ALDS [24]	Low-rank	✓	70.32	11.01	32.97
HOSA [32]	Pruning	-	64.43	43.65	91.14
DCP [8]	Pruning	-	64.22	44.75	96.60
FT [47]	Pruning	-	70.12	20.23	21.31

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.

5.2.1 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 [39]. 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 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. The supplementary material includes a further comparison of RENE with state-of-the-art vision transformer models.

5.2.2 Automatic vs. Manual Rank Selection.

We now evaluate the effectiveness of our rank search process in comparison to manual rank setting. In this experiment, we utilized pretrained ResNet18 and VGG16 models on both the CIFAR-10 and CIFAR-100 datasets. To simulate manual rank setting, we applied Tensor Train (TT) decomposition and fixed the rank uniformly across all layers. Our goal was to achieve a decomposed model with a specific percentage of the initial model's parameters, selected from the set $\{1, 5, 10, 25, 50, 75\}$. This approach allowed us to examine how varying the fixed rank impacts the performance and compression of the models.

All models were initially pre-trained on the ImageNet-1K dataset and subsequently fine-tuned for 20 epochs on the

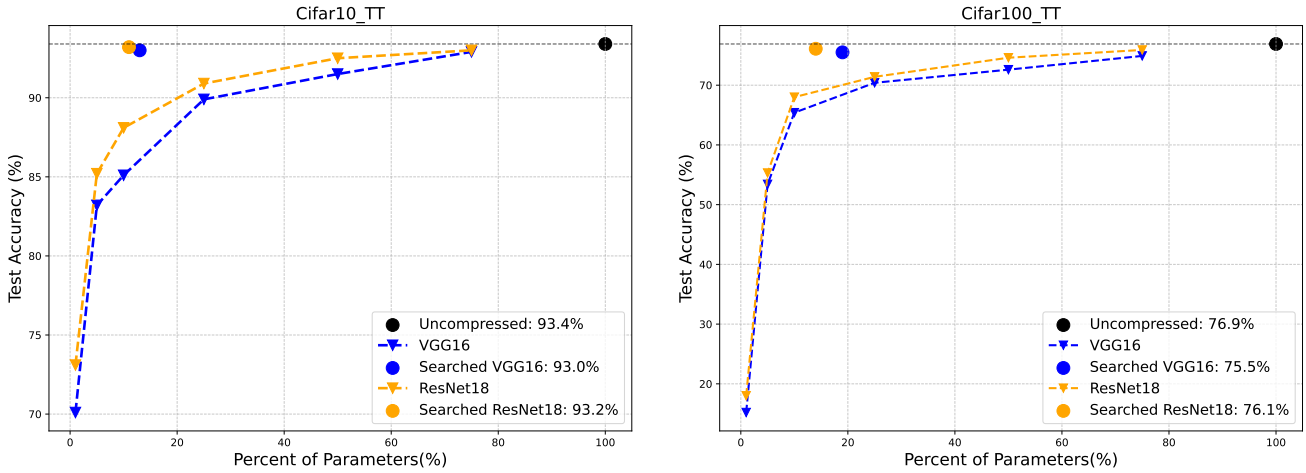

FIGURE 4 – Search vs Manual : Compression results for manual setting at different levels compression, compare to searched setting (left CIFAR10 and right CIFAR100).

CIFAR-10 and CIFAR-100 datasets. Figure 4 illustrates the results of this experiment. As depicted, increasing the number of ranks, which corresponds to increasing the percentage of parameters retained in the decomposed model, leads to improved performance for both the decomposed VGG16 and ResNet18 models. Notably, when the decomposed models retain 75% of the parameters of the original models, their performance nearly matches that of the original pretrained models.

In contrast, our approach, RENE, achieves comparable performance while compressing the models by more than 80% for both ResNet18 and VGG16 across both datasets. This highlights a key advantage of RENE : it enables the automatic selection of ranks across different layers, tailoring the compression to the importance of each layer. This adaptive approach results in a good compression rate without significant loss in performance, demonstrating that a one-size-fits-all manual rank setting is suboptimal.

These findings underscore the importance of adaptive rank selection in maintaining model performance while achieving significant compression. By allowing ranks to vary across layers, RENE effectively balances the trade-off between model compression and performance, making it a more efficient solution compared to manual rank setting.

5.2.3 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.

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.

5.2.4 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. Simi-

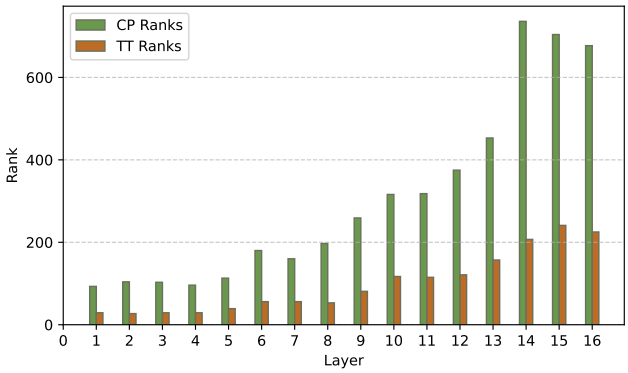

FIGURE 5 – Distribution of ranks achieved using CP and TT decompositions on ResNet-18 for the ImageNet-1K dataset.

TABLE 3 – Double compression : RENE with distillation on CIFAR-100 and ImageNet-1K.

CIFAR-100 (ResNet56 (72.34%) → ResNet20 (69.6%))			
Method	**Top-1 (%)**	**FLOPs (%)**	**Comp. rate (%)**
Distillation [36]	72.53	67.7	68.24
RENE(Teacher)	72.23	64.23	61.75
RENE(Student)	**72.46**	**89.01**	**86.54**
ImageNet-1K (ResNet34 (73.31%) → ResNet18 (69.76%))			
Method	**Top-1 (%)**	**FLOPs (%)**	**Params (%)**
Distillation [36]	71.98	50.27	46.33
RENE(Teacher)	73.23	59.91	63.46
RENE(Student)	**71.9**	**76.77**	**78.69**

larly, 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.

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.

Références

[1] Manoj Alwani, Yang Wang, and Vashisht Madhavan. Decore : Deep compression with reinforcement learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12349–12359, 2022.

[2] Hesam Amoualian, Wei Lu, Éric Gaussier, Georgios Balikas, Massih-Reza Amini, and Marianne Clausel. Topical coherence in lda-based models through induced segmentation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL 2017*, pages 1799–1809, 2017.

[3] Georgios Balikas and Massih-Reza Amini. Twise at semeval-2016 task 4 : Twitter sentiment classification. In *10th International Workshop on Semantic Evaluation*, pages 85–91, 2016.

[4] Lucas Beyer, Xiaohua Zhai, Amélie Royer, Larisa Markeeva, Rohan Anil, and Alexander Kolesnikov. Knowledge distillation : A good teacher is patient and consistent. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10925–10934, 2022.

[5] Davis Blalock, Jose Javier Gonzalez Ortiz, Jonathan Frankle, and John Guttag. What is the state of neural network pruning ? *Proceedings of machine learning and systems*, 2 :129–146, 2020.

[6] Tianxiao Cao, Lu Sun, Canh Hao Nguyen, and Hiroshi Mamitsuka. Learning low-rank tensor cores with probabilistic 0-regularized rank selection for model compression. In *Proceedings of the* 33^{rd} *International Joint Conference on Artificial Intelligence, IJCAI*, pages 3780–3788, 2024.

[7] Chi-Chih Chang, Yuan-Yao Sung, Shixing Yu, Ning-Chi Huang, Diana Marculescu, and Kai-Chiang Wu. Flora : Fine-grained low-rank architecture search for vision transformer. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 2482–2491, 2024.

[8] Christos Chatzikonstantinou, Georgios Th Papadopoulos, Kosmas Dimitropoulos, and Petros Daras. Neural network compression using higher-order statistics and auxiliary reconstruction losses. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pages 716–717, 2020.

[9] Wei Dai, Jicong Fan, Yiming Miao, and Kai Hwang. Deep learning model compression with rank reduction in tensor decomposition. *IEEE Transactions on Neural Networks and Learning Systems*, 2023.

[10] Jia Deng, Olga Russakovsky, Jonathan Krause, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. Scalable multi-label annotation. In *ACM Conference on Human Factors in Computing Systems (CHI)*, 2014.

[11] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert : Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv :1810.04805*, 2018.

[12] Ignat Domanov and Lieven De Lathauwer. Canonical polyadic decomposition of third-order tensors : Relaxed uniqueness conditions and algebraic algorithm. *Linear Algebra and its Applications*, 513 :342–375, 2017.

[13] Donald Goldfarb and Zhiwei Qin. Robust low-rank tensor recovery : Models and algorithms. *SIAM Journal on Matrix Analysis and Applications*, 35(1) :225–253, 2014.

[14] Julia Gusak, Maksym Kholiavchenko, Evgeny Ponomarev, Larisa Markeeva, Philip Blagoveschensky, Andrzej Cichocki, and Ivan Oseledets. Automated multi-stage compression of neural networks. In *Proceedings of the IEEE/CVF International Conference on Computer Vision Workshops*, pages 0–0, 2019.

[15] Christopher J Hillar and Lek-Heng Lim. Most tensor problems are np-hard. *Journal of the ACM (JACM)*, pages 1–39, 2013.

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

[17] Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora : Low-rank adaptation of large language models. *arXiv preprint arXiv :2106.09685*, 2021.

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

[19] Yong-Deok Kim, Eunhyeok Park, Sungjoo Yoo, Taelim Choi, Lu Yang, and Dongjun Shin. Compression of deep convolutional neural networks for fast and low power mobile applications. In *4th International Conference on Learning Representations, ICLR*, 2016.

[20] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

[21] Souvik Kundu, Gourav Datta, Massoud Pedram, and Peter A Beerel. 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*, pages 3953–3962, 2021.

[22] Hao Li, Asim Kadav, Igor Durdanovic, Hanan Samet, and Hans Peter Graf. Pruning filters for efficient convnets. In *International Conference on Learning Representations, ICLR*, 2017.

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

[24] Lucas Liebenwein, Alaa Maalouf, Dan Feldman, and Daniela Rus. Compressing neural networks : Towards determining the optimal layer-wise decomposition. *Advances in Neural Information Processing Systems*, 34 :5328–5344, 2021.

[25] Alexander Novikov, Dmitrii Podoprikhin, Anton Osokin, and Dmitry P Vetrov. Tensorizing neural networks. *Advances in neural information processing systems*, 28, 2015.

[26] Ivan V Oseledets. Tensor-train decomposition. *SIAM Journal on Scientific Computing*, 33(5) :2295–2317, 2011.

[27] Eunhyeok Park, Junwhan Ahn, and Sungjoo Yoo. Weighted-entropy-based quantization for deep neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5456–5464, 2017.

[28] David Patterson, Joseph Gonzalez, Quoc Le, Chen Liang, Lluis-Miquel Munguia, Daniel Rothchild, David So, Maud Texier, and Jeff Dean. Carbon emissions and large neural network training. *arXiv preprint arXiv :2104.10350*, 2021.

[29] Anh-Huy Phan, Konstantin Sobolev, Konstantin Sozykin, Dmitry Ermilov, Julia Gusak, Petr Tichavskỳ, Valeriy Glukhov, Ivan Oseledets, and Andrzej Cichocki. Stable low-rank tensor decomposition for compression of convolutional neural network. In *Computer Vision–ECCV 2020 : 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XXIX 16*, pages 522–539. Springer, 2020.

[30] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pages 8748–8763. PMLR, 2021.

[31] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2 : Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4510–4520, 2018.

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

[33] Ledyard R Tucker. Some mathematical notes on three-mode factor analysis. *Psychometrika*, 31(3) :279–311, 1966.

[34] Nick Vannieuwenhoven, Raf Vandebril, and Karl Meerbergen. A new truncation strategy for the higher-order singular value decomposition. *SIAM Journal on Scientific Computing*, 34(2) :A1027–A1052, 2012.

[35] Huan Wang, Can Qin, Yue Bai, Yulun Zhang, and Yun Fu. Recent advances on neural network pruning at initialization. *arXiv preprint arXiv :2103.06460*, 2021.

[36] Yuzhu Wang, Lechao Cheng, Manni Duan, Yongheng Wang, Zunlei Feng, and Shu Kong. Improving knowledge distillation via regularizing feature norm and direction. *arXiv preprint arXiv :2305.17007*, 2023.

[37] Bichen Wu, Xiaoliang Dai, Peizhao Zhang, Yanghan Wang, Fei Sun, Yiming Wu, Yuandong Tian, Peter Vajda, Yangqing Jia, and Kurt Keutzer. Fbnet : Hardware-aware efficient convnet design via differentiable neural architecture search. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10734–10742, 2019.

[38] Hao Wu, Patrick Judd, Xiaojie Zhang, Mikhail Isaev, and Paulius Micikevicius. Integer quantization for deep learning inference : Principles and empirical evaluation. *arXiv preprint arXiv :2004.09602*, 2020.

[39] Jinqi Xiao, Chengming Zhang, Yu Gong, Miao Yin, Yang Sui, Lizhi Xiang, Dingwen Tao, and Bo Yuan. Haloc : hardware-aware automatic low-rank compression for compact neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 10464–10472, 2023.

[40] Le Xu, Lei Cheng, Ngai Wong, and Yik-Chung Wu. Tensor train factorization under noisy and incomplete data with automatic rank estimation. *Pattern Recognition*, 141 :109650, 2023.

[41] Yinchong Yang, Denis Krompass, and Volker Tresp. Tensor-train recurrent neural networks for video classification. In *International Conference on Machine Learning*, pages 3891–3900. PMLR, 2017.

[42] Miao Yin, Huy Phan, Xiao Zang, Siyu Liao, and Bo Yuan. Batude : Budget-aware neural network compression based on tucker decomposition. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 8874–8882, 2022.

[43] Miao Yin, Yang Sui, Siyu Liao, and Bo Yuan. Towards efficient tensor decomposition-based dnn model compression with optimization framework. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10674–10683, 2021.

[44] Xiyu Yu, Tongliang Liu, Xinchao Wang, and Dacheng Tao. On compressing deep models by low rank and sparse decomposition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7370–7379, 2017.

[45] Yanfu Zhang, Shangqian Gao, and Heng Huang. Exploration and estimation for model compression. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 487–496, 2021.

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

[47] Zhuangwei Zhuang, Mingkui Tan, Bohan Zhuang, Jing Liu, Yong Guo, Qingyao Wu, Junzhou Huang, and Jinhui Zhu. Discrimination-aware channel pruning for deep neural networks. *Advances in neural information processing systems*, 31, 2018.