A MULTI-DECOMPOSITION METHOD FOR COMPRESSING LARGER AI MODELS BASED ON REINFORCEMENT LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

With the development of modern deep neural network (DNN), the scale of parameters is increasing, making it difficult to deploy models for use on resource-constrained edge devices. To address this issue, model compression is necessary, and using low-rank matrix decomposition to compress DNN models is an effective research approach. However, traditional studies on low-rank decomposition compression typically apply a single matrix decomposition method to each parameter matrix in the neural network, without considering the structural characteristics of each layer in AI models, thus failing to achieve the optimal compression effect. Therefore, this paper proposes, for the first time, a scheme for model compression using multiple decomposition methods, selecting the most suitable decomposition method for each layer in the model. However, to truly implement this approach, it is essential to balance model accuracy and compression cost. To address this, we propose a joint optimization paradigm that simultaneously optimizes model accuracy and compression rate. We also introduce a framework LMFBRL based on reinforcement learning that jointly selects the optimal decomposition method and rank. Tests were conducted on five models such as LeNet-300, ResNet-20, and Vgg-16. Compared to singly using the MF method for compressing the LeNet300 model, our approach has shown an improvement of 3.6% in compression rate and a 1.8% increase in accuracy. The test results validate the effectiveness of the algorithm proposed in this paper.

1 INTRODUCTION

With the continuous advancement of AI model capabilities, the scale of model parameters has grown exponentially. Common models like VGG and ResNet now have parameter counts reaching into the millions, necessitating significant computational resources and time to run these models. For the effective execution of these large-scale AI models, they are typically deployed on resource-rich cloud servers. However, transmitting collected data from edge devices to the cloud, processing it, and then sending back the results involves considerable transmission latency, resulting in response delays.

In the ever-expanding landscape of intelligent applications, there is an urgent need to implement edge intelligence at edge nodes to meet the demand for rapid service response in scenarios such as smart homes and autonomous driving. Deploying AI models on edge nodes allows for the quick utilization of local data for training, without the need to wait for data transmission to cloud servers. This not only reduces network latency and transmission costs but also enhances data privacy and security.

However, edge nodes have limited resources, making it challenging to deploy large-scale AI models. Existing research, as demonstrated by Cheng et al. (Denil M, 2023) (Lyu H, 2019) (Cheng Y, 2015) (Cheng Y, 2018), has shown the redundancy of AI models, providing foundational principles for model compression. Therefore, we can compress AI models and deploy the compressed versions to resource-constrained edge nodes for rapid execution.

Existing AI model compression techniques can be primarily categorized into four types: parameter quantization(Wu J, 2016)(Zhou Y, 2018), pruning(Luo J H, 2017)(Li Y, 2022), knowledge distillation(Hinton G, 2015), and low-rank decomposition(Tai C, 2015)(Wu K, 2019)(Phan A H, 2020a).

a.Parameter quantization reduces the floating-point precision of model parameters (e.g., from Float32 to Float8) to compress the model. However, when quantizing parameters to specific bit-widths, many existing training methods and hardware platforms become incompatible, necessitating the design of specialized system architectures with limited flexibility.

b.Pruning reduces the model parameter count by removing redundant weights with small values. However, it relies heavily on pre-trained models, and platforms like TensorFlow do not support sparse computation, thus hindering computational acceleration.

c.Knowledge distillation treats large AI models as teacher models, incorporating soft targets related to the teacher model as part of the total loss function to guide the training of student models, resulting in compressed models. However, its assumptions about student models are overly restrictive, limiting its applicability.

d.Low-rank decomposition utilizes techniques to decompose weight matrices/tensors of AI models into smaller scale low-rank matrices/tensors, achieving parameter compression. This method does not require specialized hardware support, can be used across various platforms, and offers flexibility in compression levels by adjusting the rank of the decomposition matrices.

Based on the analysis of the four AI model compression methods mentioned above, this paper adopts the approach of low-rank decomposition due to its advantages of low hardware requirements and flexible compression.

1.1 CHALLENGE

Current research on AI model compression includes numerous low-rank decomposition methods, primarily categorized into low-rank matrix decomposition and low-rank tensor decomposition (Wang C, 2024) (Tang J, 2022) (Liu Y, 2022). Two-dimensional matrix decomposition is simpler and faster than high-dimensional tensor decomposition, with Singular value decomposition (SVD) and Full-rank matrix factorization (MF) being the main methods. Most existing research employs a single decomposition method for all neural network layers. For instance, Hsu et al. (Hsu, 2017) used SVD for all layers, while Winata et al. (Winata, 2015) used MF. These methods have been effective for model compression.

However, our experiments reveal that different neural network layers have distinct structural characteristics, and various decomposition methods have different capacities for representing these structures. This limitation becomes apparent when using a single decomposition method. Our tests show that applying different decomposition methods to different network layers yields varying compression effectiveness, with the optimal method differing for each layer (detailed analysis in Section 3).

To address this issue, we explore using multiple decomposition methods for model compression, selecting the most suitable method for each layer. However, achieving this multi-decomposition approach requires addressing two key challenges:

a.Balancing model accuracy and compression cost: Compressing a model often reduces accuracy. Striving for higher compression can result in progressively lower accuracy. Existing methods tend to focus on one aspect (Xu Y, 2020) (Yang H, 2020), leading to either high compression cost and low accuracy or low compression cost and high accuracy. Balancing the trade-off between accuracy and compression cost to achieve optimal outcomes for both is extremely challenging.

b.Jointly selecting the decomposition method and the rank r: The performance of low-rank decomposition compression is influenced by the chosen rank r. Existing methods determine the rank based on a single predetermined decomposition method. Our approach, involving multiple decomposition methods, requires the joint selection of both the decomposition method and the rank, which is highly challenging.

1.2 CONTRIBUTION

To address these challenges and achieve optimal model compression, this paper proposes the Low-Rank Matrix Factorization Algorithm Based On Reinforcement Learning (LMFBRL). This method selects the optimal decomposition method for each layer, determines the appropriate rank r, balances the trade-off between compression ratio and accuracy, and achieves optimal results. The contributions are as follows:

a.Joint Optimization of Model Accuracy and Compression Cost: This paper introduces a paradigm that jointly considers model accuracy and cost functions in the context of multiple decomposition methods. By constructing expressions for both and jointly optimizing them, we achieve an optimal balance between model accuracy and compression, resulting in the best possible solution for model compression.

b.Alternating Training for Compressed Models Without Retraining: We propose an alternating optimization algorithm that iteratively performs parameter learning and model compression. Initially, model parameters are updated using the model's loss function and a compression parameter matrix. Subsequently, each network layer's parameter matrix undergoes decomposition and compression. This alternating process is repeated through successive training iterations until the final training epoch is reached. Notably, the resulting compressed model does not require retraining after the alternating optimization process is completed.

c.Joint Selection of Decomposition Method and Rank Based on Reinforcement Learning: Leveraging reinforcement learning, we employ the Q-learning algorithm to establish a Q-value matrix based on two different decomposition methods. Each Q-value is determined by the model's loss value and an optimal rank selection function. Within the reinforcement learning framework, a selection function is constructed to choose the decomposition method by comparing Q-values. This approach not only solves for the Q-values to select the decomposition method but also determines the optimal rank simultaneously, achieving the goal of jointly selecting the best decomposition method and rank.

d.Extensive Experiment: We conducted experiments using five models: LeNet-300, LeNet-5, ResNet-20, ResNet-32, and Vgg-16, tested on two datasets, MNIST and CIFAR-10. The experimental results demonstrate that, compared to using a single fixed decomposition method for model compression, our LMFBRL approach achieves better compression rates with minimal accuracy loss. Specifically, LeNet-300 achieved a compression rate of 89.5% and an accuracy of 96.2%. When applying MF decomposition solely to LeNet-300, the compression rate was 83.6% with an accuracy of 95.2%. In comparison, our LMFBRL approach resulted in a 3.6% improvement in compression rate and a 1.8% increase in accuracy. Test results confirm the effectiveness of our algorithm.

2 RELATED WORK

For an L-layer neural network model, the output is represented as $O_i = \Psi(W_i o_{i-1} - b_i)$, where $i \in \{1, 2, \cdots, L\}$. Here, O_i denotes the output of the i-th layer, $x_i = o_{i-1}$ is the input to the i-th layer, $\Psi(*)$ represents the activation function, and b_i is the bias term. The principle of low-rank decomposition involves decomposing the parameter matrix W of each layer into smaller matrices. The main methods (MF and SVD) are discussed in Section A.1 and Section A.2.

In low-rank decomposition-based AI compression, performance is influenced by the chosen rank r, affecting both compression rate and accuracy. Thus, selecting the optimal rank is crucial.

Some studies use a fixed rank r for each layer's decomposition. For example, Kim et al. (Kim Y D, 2015) use variational Bayesian matrix decomposition to select ranks for CNNs on mobile devices, then decompose and fine-tune the model. Kim et al. (Kim H, 2019) propose an effective method considering the entire network to choose the correct rank configuration, using new accuracy metrics non-iteratively. These fixed-rank methods can achieve high compression rates but often compromise accuracy, leading to poor overall performance.

To address these issues, some researchers propose automated rank selection using performance metrics or heuristics. Phan et al. (Phan A H, 2020b) introduced SLR low-rank decomposition, using positive definite tensors to reduce parameters and costs. Xiao et al. (Xiao J, 2023) proposed HALOC, a hardware-aware framework for automatic rank determination. Other studies (Kim Y D, 2015) (Kim H, 2019) (Acharya A, 2019) (Tai C, 2015) (Lebedev V, 2014) also use rank selection to improve compression but focus solely on rank r without optimizing model accuracy. It is not possible to achieve optimal values for both compression rate and accuracy.

Moreover, the aforementioned studies all employ the same decomposition method for compressing neural networks. However, our analysis shows that the optimal decomposition method varies for different layers of the neural network, and also differs based on the structure of those layers. As a result, a fixed decomposition scheme does not yield the best compression performance.

To solve these issues, this paper introduces a joint optimization paradigm for model accuracy and compression cost, ensuring both are optimized. Inspired by reinforcement learning, we construct a framework (LMFBRL) for joint selection of decomposition methods and ranks. This framework selects the best method and optimal rank for each layer, achieving the best possible model compression.

3 Experimental Studies: Different Decomposition Methods for Different Network Layers

This paper makes a novel discovery that, due to the differing structural characteristics of the layers in AI models and the varying representation capabilities of two decomposition methods, using a single decomposition approach cannot achieve optimal model compression. The following sections will introduce this finding and analyze its impact on compression performance.

As shown in Fig. 1, there is a variety of types of neural network layers, primarily including fully connected layers and convolutional layers, along with others such as pooling layers and flatten layers. The decomposition methods need to focus on compressing the parameters of fully connected and convolutional layers. Convolutional layers primarily filter and extract input features at the front end, while fully connected layers perform feature classification at the end.

Additionally, the two low-rank decomposition methods, MF and SVD, operate differently (see Appendix 1). Consequently, their decomposition and compression effects on the same neural network layer also differ. We will validate these findings through experiments by comparing the results of compression performance tests.

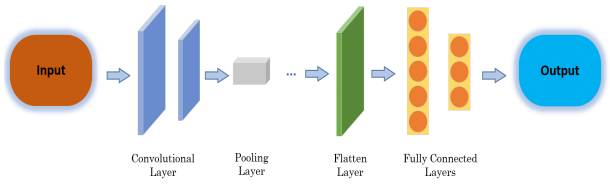

Figure 1: The structure of deep neural network.

For our experiments, we selected the VGG-16 model and focused on compressing the 5th convolutional layer and the 14th fully connected layer. We applied both MF and SVD methods to these layers. We then compared and analyzed the compression performance of these two methods. The test results are presented in Table 1.

In Table 1, the "Method" column indicates the decomposition method used for the 5th and 14th layers. For example, MF-SVD means MF decomposition was used for the 5th layer and SVD method for the 14th layer. The parameters Params5 and Params14 represent the number of parameters in these layers. The metrics ACC, ρ_{params}, and τ_{params} denote the model's accuracy, parameter compression ratio, and parameter compression performance, respectively. The higher their values, the better the compression effect. Detailed explanations of these metrics can be found in Section 8.

Table 1: Compression of Network Layers Using Two Decomposition Methods

Method	Params5	Params14	ACC	ρ_{params}	τ_{params}
Original	294912	262144	89.1%	0	-
MF-MF	35200	25600	86.9%	86.3%	39.2
SVD-SVD	66176	10240	85.9%	89.1%	27.8
MF-SVD	46464	33792	**87.2%**	85.6%	**45**
SVD-MF	16896	24576	85.7%	**92.6%**	27.2

By comparing the τ_{params} values, we see that MF-MF with τ_{params}=39.2 is greater than SVD-SVD with τ_{params}=27.8, indicating that the MF-MF scheme has better compression performance. Further comparing MF-MF's τ_{params}=39.2 with MF-SVD's τ_{params}=45, we find that MF-SVD has superior compression performance. Additionally, comparing MF-SVD's τ_{params}=45 with SVD-MF's τ_{params}=27.2, MF-SVD still proves to be better. This suggests that the 5th layer is more suited to MF decomposition, while the 14th layer benefits more from SVD method.

By comparing the parameter compression performance τ_{params} across the four schemes in the table, we conclude that the MF-SVD scheme offers the best compression performance. **This indicates that, due to the different structural characteristics of network**

layers and the varying representational capacities of decomposition methods, selecting the most suitable decomposition method for each layer is essential for achieving optimal model compression.

4 PROBLEM FORMULATION

In response to the issue of single decomposition methods failing to consider the structural characteristics of various network layers in AI models, this paper proposes the LMFBRL approach. It employs multiple decomposition methods for model compression, selecting the most suitable method for each layer in the model to achieve optimal compression results.

For a neural network model with L layers, the weight matrices of the model are represented as $W = \{W_i\} = \{W_1, W_2, \ldots, W_L\}$, where $W_i \in \mathbb{R}^{m_i \times n_i}$, and $i = \{1, 2, \ldots, L\}$ denotes the index of the i-th layer of the model. Considering the interplay between compression ratio and model accuracy mentioned earlier, we have formulated the following problem paradigm to balance the relationship between the two:

$$\begin{aligned} &\min_{W} L\left(W\right) + \mu C\left(r\left(W\right)\right) \\ &\text{s.t. rank}\left(W_i\right) \leq R_i, i = 1, 2, \ldots, L. \end{aligned} \tag{1}$$

In the above Eq. (1), $L\left(W\right)$ represents the loss function, which indicates the model's accuracy, while $C\left(r\left(W\right)\right)$ represents the cost function, which denotes the number of parameters, thereby approximating the compression ratio of the model. In this problem paradigm, R_i denotes the maximum rank of each layer's parameter matrix, where $R_i \leq \min\{m_i, n_i\}$. Here, μ serves as a hyperparameter to balance the importance between the compression ratio and model accuracy. μ belongs to the interval $[0, 1]$. As μ approaches 0, the model prioritizes accuracy improvement, whereas as μ approaches 1, it prioritizes increasing the model's compression ratio.

This study involves selecting between two decomposition methods: MF and SVD. Therefore, the cost function $C(r(W))$ in Eq. (1) must account for different decomposition method conditions. For each layer, the cost function presents two options, as shown in Eq. (2):

$$\begin{aligned} C\left(r\left(W\right)\right) &= \sum_{i=1}^{L} \text{Best}\{C_{MF}\left(r\left(W_i\right)\right), C_{SVD}\left(r\left(W_i\right)\right)\} \\ &\text{s.t. rank}\left(W_i\right) \leq R_i, i = 1, 2, \ldots, L. \end{aligned} \tag{2}$$

In Eq. (2) above, for the cost function of each network layer, we derive the cost function $C_{MF}(r(W_i))$ from MF decomposition and $C_{SVD}(r(W_i))$ from SVD method. By utilizing the Best function, we select the optimal cost function from these two decomposition methods to determine the chosen decomposition method. The principles of the two decomposition methods and their cost functions can be found in the Appendix A.1 and A.2.

To adaptively select the optimal decomposition method for each layer of the AI model, we combine the cost functions of the two decomposition methods: $C_{MF}\left(r\left(W_i\right)\right) = r_i \cdot (m_i + n_i)$ and $C_{SVD}\left(r\left(W_i\right)\right) = r_i \cdot (m_i + r_i + n_i)$. We then rewrite Eq.(2) as follows:

$$\begin{aligned} &\min_{W, \phi_i, \omega_i, r_i} L\left(W\right) + \mu \sum_{i=1}^{L} \{\phi_i\left(r_i \cdot (m_i + n_i)\right) + \omega_i\left(r_i \cdot (m_i + r_i + n_i)\right)\} \\ &\text{s.t. } \phi_i, \omega_i \in \{0, 1\}, \phi_i + \omega_i = 1, r_i = \text{rank}\left(W_i\right), \text{rank}\left(W_i\right) \leq R_i, i = 1, 2, \ldots, L. \end{aligned} \tag{3}$$

where ϕ_i and ω_i are two binary indicator parameters for the i-th layer of AI model, and ϕ_i and ω_i can only take 0 or 1. Specifically, when $\phi_i = 1$ and $\omega_i = 0$, it indicates that the i-th network layer selects the MF decomposition. Conversely, when $\phi_i = 0$ and $\omega_i = 1$, it indicates that the i-th network layer selects the SVD method. Due to $\phi_i + \omega_i = 1$, each layer can adopt only one decomposition method.

5 OVERVIEW OF SOLUTION

To solve the problem (3), we need to optimize and solve for parameters W, ϕ_i, ω_i, and r_i. To address this multivariable problem, we first introduce intermediate variables $X = \{X_1, X_2, \ldots, X_L\}$, and impose the constraint $\{X_i\} = \{W_i\}$. We then reformulate Eq. (4) as a constrained optimization problem:

$$\begin{aligned} &\min_{W, \phi_i, \omega_i, r_i} L\left(W\right) + \mu \sum_{i=1}^{L} \{\phi_i\left(r_i \cdot (m_i + n_i)\right) + \omega_i\left(r_i \cdot (m_i + r_i + n_i)\right)\} \\ &\text{s.t. } \phi_i, \omega_i \in \{0, 1\}, \phi_i + \omega_i = 1, r_i = rank\left(W_i\right), X_i = W_i, rank\left(W_i\right) \leq R_i, i = 1, 2, \ldots, L. \end{aligned} \tag{4}$$

Based on the augmented Lagrangian method, we can transform the above problem as follows:

$$\begin{aligned} &\min_{W, \phi_i, \omega_i, r_i, X_i} L\left(W\right) + \mu \sum_{i=1}^{L} \{\phi_i\left(r_i \cdot (m_i + n_i)\right) + \omega_i\left(r_i \cdot (m_i + r_i + n_i)\right)\} + \frac{\beta}{2} \sum_{i=1}^{L} ||W_i - X_i - \frac{1}{\beta}\gamma_i||^2 \\ &\text{s.t. } \phi_i, \omega_i \in \{0, 1\}, \phi_i + \omega_i = 1, r_i = rank\left(X_i\right), rank\left(X_i\right) \leq R_i, i = 1, 2, \ldots, L. \end{aligned} \tag{5}$$

where β is the penalty parameter, which increases with the step of iterations in the model, $\beta \in (\beta_1 < \beta_2 < \cdots)$. Detailed analysis of this approach can be found in Section 5.2. $\{\gamma_1, \gamma_2, \cdots, \gamma_L\}$ are the vector of Lagrange multipliers, with dimensions matching those of the weight matrixs $\{W_1, W_2, \ldots, W_L\}$, and it is updated during each iteration.

To solve Eq. (5), we have designed an alternating optimization method by alternately optimizing W, ϕ_i, ω_i, r_i and X_i. So Eq. (5) is divided into two sub-problems to be solved alternately. The first sub-problem focuses on updating the model parameters, which is suggested in section 5.1. The second sub-problem involves selecting the decomposition method and determining the optimal rank for decomposition, which is suggested in section 6. The detailed solution method is outlined in Algorithm 1.

Algorithm 1 Alternating solution algorithm for model parameters updating and compression selecting

0: **Initialize:** The pre-trained model parameters W, the selected compression ranks r, the compressed model parameters X, and the Lagrange multipliers γ are all initialized to zero.
1: **for** $\beta = \beta_1 < \beta_2 < \cdots$ **do**
2: Sub-problem 1: Model Parameters Updating
3: $W \leftarrow \arg\min_W L(W) + \frac{\beta}{2} \sum_{i=1}^{L} ||W_i - X_i - \frac{1}{\beta}\gamma_i||^2$
4:
5: Sub-problem 2: Joint Selection of Decomposition and Rank r
6: $\phi_i, \omega_i, r_i, X_i \leftarrow \arg\min_{\phi_i, \omega_i, r_i, X_i} \mu \sum_{i=1}^{L} \{\phi_i (r_i \cdot (m_i + n_i)) + \omega_i (r_i \cdot (m_i + r_i + n_i))\} + \frac{\beta}{2} \sum_{i=1}^{L} ||W_i - X_i - \frac{1}{\beta}\gamma_i||^2$
7: **end for**

5.1 The Optimization Of Parameter W

The first sub-problem involves updating the learning of model parameters. We need to solve the objective function in line 3 of Algorithm 1, where the compressed matrix X_i corresponding to the parameter matrix W_i of each network layer is a known condition. The SGD gradient descent algorithm is utilized to solve based on the model's loss and the constraints imposed by the compression matrix on the original parameter matrix. The solving formula is as follows:

$$W^* = W - \alpha \cdot \frac{\partial \left(L(W) + \frac{\beta}{2} \sum_{i=1}^{L} ||W_i - X_i - \frac{1}{\beta}\gamma_i||^2 \right)}{\partial W}. \tag{6}$$

In the above Eq. (6), W^* represents the updated parameters, W denotes the original parameters, α is the learning rate, and the fraction following it represents the partial derivative with respect to parameter W. This equation completes the update of model parameters, constituting the entire learning and updating process of the model. Next, we will elaborate on the solution to the second subproblem in section 6.

5.2 The Update Of Parameter β

The loop statement in the 1 line of Algorithm 1 indicates that as the number of iterations increases throughout the process, we gradually increase our penalty factor β. In line 3 of Algorithm 1, the loss function $L(W)$ in the objective function for parameter updates is used to improve model accuracy. The subsequent regularization term $||W_i - X_i - \frac{1}{\beta}\gamma_i||^2$ constrains the parameter matrix W_i to approximate the low-rank compressed matrix X_i. This makes the parameter matrix more sparse, thereby enhancing the model's compression rate. Consequently, when the value of β is small, the loss function $L(W)$ has a greater impact, and the model training focuses more on improving accuracy. As the value of β increases, the influence of the regularization term becomes more significant, and the training emphasizes enhancing the compression rate. Therefore, by the end of the training, the model achieves an optimal balance between accuracy and compression rate.

6 Reinforcement Learning Framework For Selection Of Decomposition Method And Rank

This section will provide a detailed analysis of the second sub-problem of Algorithm 1. In order to solve for the parameters ϕ_i, ω_i, r_i and X_i, which allows us to select the optimal decomposition method for each network layer and determine the best rank to obtain the optimal compressed matrix. The objective function for the compression process is defined as follows:

$$\begin{aligned} &\min_{\phi_i, \omega_i, r_i, X_i} \mu \sum_{i=1}^{L} \{\phi_i (r_i \cdot (m_i + n_i)) + \omega_i (r_i \cdot (m_i + r_i + n_i))\} + \frac{\beta}{2} \sum_{i=1}^{L} ||W_i - X_i - \frac{1}{\beta}\gamma_i||^2 \\ &\text{s.t. } \phi_i, \omega_i \in \{0, 1\}, \phi_i + \omega_i = 1, r_i = rank(X_i), rank(X_i) \leq R_i, i = 1, 2, \ldots, L. \end{aligned} \tag{7}$$

To solve Eq. (7), we propose a reinforcement learning-based framework for multi-decomposition method selection (LMFBRL). Next, we will provide a detailed explanation of this framework. For a detailed explanation of reinforcement learning, please refer to Appendix A.3.

6.1 CALCULATING THE REWARD VALUES Q UNDER DIFFERENT DECOMPOSITIONS

In the reinforcement learning framework of this paper, it is necessary to make choices for the decomposition methods of each layer, and the criterion for selection is the reward value Q of the corresponding decomposition method, we employ the Q-learning method. So we construct a reward matrix $\Omega \in \mathbb{R}^{2\times L}$, defined as Table 2.

Table 2: Reward matrix Ω

Method \ Layer	1	2	$\cdots$	L
MF	$Q_{1,mf}$	$Q_{2,mf}$	$\cdots$	$Q_{L,mf}$
SVD	$Q_{1,svd}$	$Q_{2,svd}$	$\cdots$	$Q_{L,svd}$

Our objective is to maximize the sum of the reward values for each layer, $\mathbb{Q} = \sum_{i=1}^{L} \max\{Q_{i,mf}, Q_{i,svd}\}$, where $Q_{i,mf}$ represents the value obtained by selecting the MF decomposition method for the i-th network layer, and $Q_{i,svd}$ represents the value obtained by selecting the SVD method for the i-th network layer. Here, we employ a greedy algorithm, selecting the decomposition method for each layer that yields the maximum gain until the final layer, thereby achieving the overall maximum reward value $\mathbb{Q}$.

To simultaneously select the decomposition method and the corresponding optimal rank, this paper cleverly sets the Q values for each decomposition method as the results obtained from solving the corresponding rank selection functions. In other words, solving Eq. (7) yields the optimal ranks for the two decomposition methods. Therefore, the rank selection function and Q value of each decomposition method are equivalent.

When we choose the MF method, the rank selection function for the MF method in the i-th network layer is equivalent to $Q_{i,mf}$ solution. From Eq. (7), we obtain the following:

$$\begin{aligned} Q_{i,mf} &= \mu \cdot r_i \cdot (m_i + n_i) + \frac{\beta}{2}||W_i - X_{i,mf} - \frac{1}{\beta}\gamma_i||^2 \\ \text{s.t. } r_i, X_i \leftarrow min_{r_i,X_i} & r_i \cdot (m_i + n_i) + \frac{\beta}{2}||W_i - X_{i,mf} - \frac{1}{\beta}\gamma_i||^2, r_i = rank\left(X_{i,mf}\right), rank\left(X_{i,mf}\right) \le R_i, i = 1, 2, \ldots, L. \end{aligned} \tag{8}$$

We observe from Eq. (8) that as the rank r_i decreases, the number of parameters in the decomposition matrix reduces. However, the error between the MF decomposition matrix $X_{i,mf}$ and the weight matrix W_i increases for the i-th layer of the model. Hence, it is necessary to find an optimal solution $r_{i,\text{best}}$ that minimizes the result, which is our objective. This requires traverse all the ranks $\{r_i\} = \{r_{i,1}, r_{i,2}, \ldots\}$.

Alternatively, when we choose the SVD method, we present the following formula as the solution for the rank selection function of SVD method from Eq. (7), which is equivalent to $Q_{i,svd}$ solution:

$$\begin{aligned} Q_{i,svd} &= \mu \cdot r_i \cdot (m_i + r_i + n_i) + \frac{\beta}{2}||W_i - X_{i,svd} - \frac{1}{\beta}\gamma_i||^2 \\ \text{s.t. } r_i, X_i \leftarrow \min_{r_i,X_i} & r_i \cdot (m_i + n_i) + \frac{\beta}{2}||W_i - X_{i,svd} - \frac{1}{\beta}\gamma_i||^2, r_i = rank\left(X_{i,svd}\right), rank\left(X_{i,svd}\right) \le R_i, i = 1, 2, \ldots, L. \end{aligned} \tag{9}$$

When solving Eq. (9) for the SVD method, according to (Golub and van Loan., 2012) and based on the Eckhart-Young theorem, $||W_i - X_{i,svd} - \frac{1}{\beta}\gamma_i||^2 \iff ||\sum_{l=r_i+1}^{R_i} s_{il}^2 - \frac{1}{\beta}\gamma_i^2||$. So Eq. (9) is equivalent to:

$$\begin{aligned} Q_{i,svd} &= \mu \cdot r_i \cdot (m_i + r_i + n_i) + \frac{\beta}{2}||\sum_{l=r_i+1}^{R_i} s_{il}^2 - \frac{1}{\beta}\gamma_i^2|| \\ \text{s.t. } r_i, X_i \leftarrow \min_{r_i,X_i} & r_i \cdot (m_i + n_i) + \frac{\beta}{2}||\sum_{l=r_i+1}^{R_i} s_{il}^2 - \frac{1}{\beta}\gamma_i^2||, r_i = rank\left(X_{i,svd}\right), rank\left(X_{i,svd}\right) \le R_i, i = 1, 2, \ldots, L. \end{aligned} \tag{10}$$

To solve Eq. (10), similar to the method for solving the MF problem, we need to iterate through all the ranks to find the optimal rank that minimizes the result.

6.2 SELECTION OF DECOMPOSITION METHODS

By solving the Eq. (8) and Eq. (10), we obtain both the reward value Q_i and the rank r_i of the i-th layer. Therefore, to select the optimal decomposition method for each layer of the model, we define the method selection function $E(\phi, \omega)$, which derives the optimal decomposition method for the model based on the reward value Q, is expressed as follows:

$$E(\varphi, \omega) = \bigcup_{i=1}^{L} \begin{cases} (\varphi_i, \omega_i) = \{1, 0\}, \mathrm{Q}_{i,mf} \ge \mathrm{Q}_{i,svd}, \\ (\varphi_i, \omega_i) = \{0, 1\}, \mathrm{Q}_{i,mf} < \mathrm{Q}_{i,svd}. \end{cases} \tag{11}$$

Combining Eq. (11) with the previous Eq. (7), the best compressed matrix for the i-th layer is $X_{i,\text{best}} = E(\phi_i, \omega_i) \cdot \{X_{i,mf}, X_{i,svd}\}$, obtained by multiplying the decomposition matrices.

As show in Fig. 2, the parameter matrices W_i of each layer of the trained model are decomposed into the product form of decomposition matrices according to the recorded selection method. The first layer selects the SVD method, so the corresponding weight matrix W_1

is decomposed into the product of matrices U_1, S_1 and V_1. The second layer selects the MF method, so the corresponding weight matrix W_2 is decomposed into the product of matrices U_2 and V_2. The subsequent network layers are decomposed into corresponding compression structures according to the corresponding decomposition method selected.

In summary, we design a reinforcement learning-based framework that simultaneously selects the optimal decomposition method and rank r. The reward values $Q_{i,mf}$ and $Q_{i,svd}$ for the two decomposition methods are determined by solving the corresponding rank selection functions. By comparing the obtained reward values $Q_{i,mf}$ and $Q_{i,svd}$, the optimal decomposition method is identified using the method selection function $E(\phi, \omega)$. Thus, the optimal decomposition method and best rank for the network layers in the model can be derived simultaneously.

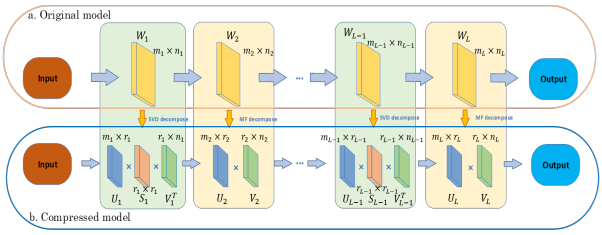

Figure 2: The compressed model based on LMFBRL.

7 Completed Solution

This section provides a comprehensive analysis of the LMFBRL algorithm designed in this paper. The algorithm adopts an iterative training approach, termed "update before compression selection," which involves training the model parameters first and then compressing them using a reinforcement learning-based decomposition algorithm across all layers. The compressed matrices obtained are then used to train the entire model parameters, and this process is iteratively repeated until the final training round. The detailed procedure is outlined in Algorithm 2.

Through the aforementioned iterative method of training before compression, as the number of iterations increases, we adjust our penalty factor β towards infinity, causing the model parameter matrix W to gradually approach the low-rank compressed matrix X, becoming more low-rank. After completing all training, based on the recorded optimal decomposition methods for each layer, the trained model parameter matrix W is decomposed into various decomposition matrices, thus achieving model compression.

The entire model training method employs the SGD (Stochastic Gradient Descent) algorithm, and the pre-trained model is incorporated into the LMBFRL algorithm for training. Lines 1 to 24 of the Algorithm 2 represent the alternating optimization scheme designed in this paper. Line 2 involves the training process for the entire model parameters, where W represents the original uncompressed model Weight matrix with rank R_i. During the learning steps of the algorithm, parameter updates are performed, while X represents the compressed low-rank matrix with rank $r_i \leq R_i$, obtained during the compression selection steps of the algorithm.

Lines 3 to 21 of the Algorithm 2 outline the model compression process, wherein all network layers of the model undergo decomposition and compression. During the reinforcement learning process, the selection of decomposition methods for each layer is determined by comparing the utility function values Q obtained from different decomposition methods. The selection function E chooses the decomposition method with the highest utility and simultaneously determines the selected rank r. The resulting compressed matrix X is then used in the parameter update steps. This process is iteratively repeated until the final round, where the trained model is decomposed according to the selected method and rank r to obtain the final compressed model.

The final compressed model obtained in this study does not require retraining to restore accuracy, saving a significant amount of time.

8 Experiment

Our experiment tests five models: LeNet-300, LeNet-5, ResNet-20, ResNet-32, and Vgg-16. Six algorithms are employed: LMFBRL-Selected, SVD-Selected, MF-Selected, LMFBRL-Fixed, SVD-Fixed, and MF-Fixed. Two datasets are utilized: MNIST and CIFAR-10. The parameter metrics consist of seven indicators: Params, FLOPs, Accuracy, ρ_{params}, ρ_{FLOPs}, τ_{params}, and τ_{FLOPs}. For detailed information, please refer to the Appendix A.4.

Our experimental setup includes an NVIDIA GeForce RTX 4070 GPU, 32GB of memory, running on Windows 11. We used Python version 3.7 and PyTorch version 1.31.0. Experiments were conducted on five AI models using the two datasets and five algorithms.

In our experiments, the parameter μ was set to 1×10^{-6} and 1×10^{-8}. The algorithm was trained for 40 iterations, with an initial β value of 1.09. For the SGD algorithm, the learning rate was set to 0.09, the decay rate to 0.98, and the momentum to 0.9. We used the MNIST and CIFAR10 datasets for training and testing.

8.1 Verification of the LMFBRL Framework

Comparison with Fixed Decomposition Methods. This experiment aims to validate the effectiveness of the multi-decomposition method selection frame-

Algorithm 2 Low rank matrix factorization algorithm based on reinforcement learning

0: **Initialize:** The pre-trained model parameters W, the selected compression ranks r, the compressed model parameters X, and the Lagrange multipliers γ are all initialized to zero.
for $\beta = \beta_1 < \beta_2 < \cdots$ **do**
2: Update W by Eq. (6)
for $i = 1, \ldots, L$ **do**
4: Matrix decomposition process
Perform SVD on the i-th layer:
6: Optimize $r_{i,svd}, \mathrm{X}_{i,svd}$ by Eq. (10)
Perform MF on the i-th layer:
8: Optimize $r_{i,mf}, \mathrm{X}_{i,mf}$ by Eq. (8)
10: Choose best matrix decomposition method:
Calculate $\mathrm{Q}_{i,svd}$ by (Eq. 10)
12: Calculate $\mathrm{Q}_{i,mf}$ by Eq. (8)
Optimize (ϕ_i, ω_i) by Eq. (11)
14: $X_{i,\text{best}} = (\phi_i, \omega_i) \cdot \{X_{i,\text{mf}}, X_{i,\text{svd}}\}$
$\gamma_i \leftarrow \gamma_i - \beta\,(W_i - X_{i,\text{best}})$
16: **end for**
If $\|W - X_{best}\| \leq \varepsilon$
18: break
end for
20: return W, X_{best}, r_{best}

Table 3: Comparison of Model Compression Methods

Model	Method	Params	FLOPs	ACC	ρ_{params}	ρ_{FLOPs}	τ_{params}	τ_{FLOPs}
LeNet-300	Original	266610	1066030	97.8%	0	0	-	-
	SVD-Selected	25253	100605	95.5%	**90.5%**	**90.5%**	39.4	39.4
	MF-Selected	43782	174718	95.2%	83.6%	83.6%	32.1	32.1
	LMFBRL-Selected	33037	131122	**97%**	87.6%	87.7%	**109.5**	**109.6**
LeNet-5	Original	431080	9173530	98.3%	0	0	-	-
	SVD-Selected	15688	2336890	94.5%	**96.4%**	**74.5%**	25.4	19.6
	MF-Selected	54010	7348650	97.3%	87.5%	19.9%	87.5	19.9
	LMFBRL-Selected	30742	4293910	**97.8%**	92.7%	53.2	**185.4**	**106.4**
ResNet-20	Original	269722	162204190	87.4%	0	0	-	-
	SVD-Selected	118522	29203072	85.9%	**56.1%**	**82%**	37.4	54.7
	MF-Selected	254957	57395214	**87%**	5.5%	64.6%	13.7	161.5
	LMFBRL-Selected	162653	29142656	86.9%	39.7%	**82%**	**79.4**	**164**
ResNet-32	Original	464154	275450398	88.6%	0	0	-	-
	SVD-Selected	180986	46776960	86.2%	**61%**	**83%**	25.4	34.6
	MF-Selected	438893	92463312	87.9%	5.4%	66.4%	7.9	94.9
	LMFBRL-Selected	285484	50735744	**88%**	38.5%	81.6%	**64.2**	**136**
Vgg-16	Original	14990922	1253856798	89.1%	0	0	-	-
	SVD-Selected	543223	215234798	83.9%	96.4%	82.8%	18.5	15.9
	MF-Selected	485689	165246814	65.4%	**96.8%**	86.8%	4.1	3.7
	LMFBRL-Selected	647133	14990922	**84.1%**	95.7%	**98.8%**	**19.1**	**19.8**

work based on reinforcement learning, referred to as LMFBRL. The algorithms compared in this study are LMFBRL-Selected, SVD-Selected, and MF-Selected, and the models tested include five architectures: LeNet-300, LeNet-5, ResNet-20, ResNet-32, and Vgg-16.

The LMFBRL-Selected algorithm uses both SVD and MF methods and selects the optimal decomposition method for each layer of the AI models. In contrast, SVD-Selected applies a fixed SVD method to decompose the AI models, and MF-Selected uses a fixed MF method. The results of the experiments are presented in Table 3, where the metric accuracy(ACC) represents the precision of the compressed models. A higher value indicates greater accuracy in the compressed model, with the highest values highlighted in bold.

From the table, we observe that the LMFBRL-Selected algorithm achieves the highest accuracy for four out of the five models, except for ResNet-20. For the ResNet-20 model, the accuracy of the LMFBRL-Selected algorithm is only 0.1% lower than the highest-performing algorithm, MF-Selected. Therefore, compared to the other two algorithms, LMFBRL-Selected generally produces the best accuracy for the compressed models.

This analysis demonstrates the high effectiveness of the LMFBRL framework. Both SVD-Selected and MF-Selected rely on fixed decomposition methods, which cannot account for the structural characteristics of the network layers in AI models. In contrast, the LMFBRL-Selected algorithm, based on reinforcement learning, **selects the most appropriate decomposition method for each layer**, thus achieving optimal compression accuracy.

8.2 Verification of the Joint Paradigm of Model Accuracy and Compression Cost

Comparison with Non-Joint Optimization Methods. This experiment aims to demonstrate the effectiveness of the joint paradigm of model accuracy and compression cost proposed in this paper. The algorithms compared in this study are LMFBRL-Selected, SVD-Selected, and MF-Selected, and the models tested include five architectures: LeNet-300, LeNet-5, ResNet-20, ResNet-32, and Vgg-16.

The objective function of the LMFBRL-Selected algorithm is based on the joint paradigm of model accuracy and compression cost, as proposed in this paper. In contrast, the other two algorithms, SVD-Selected and MF-Selected, use an objective function that optimizes only the compression cost. The experimental results are presented in Table 3, where the parameters τ_{params} and τ_{FLOPs} represent the compression performance in terms of computation and parameter reduction, respectively. These are expressed as the ratio of compression rate to accuracy loss, with higher values indicating better compression performance.

From the Table 3, we observe that the LMFBRL-Selected algorithm achieves the highest values for both τ_{params} and τ_{FLOPs} across all models. This indicates that, compared to the other two algorithms, LMFBRL-Selected delivers optimal performance in both accuracy and compression rate.

This analysis demonstrates the high effectiveness of the proposed joint optimization paradigm for model accuracy and compression cost. In contrast, the SVD-Selected and MF-Selected algorithms focus solely on optimizing the compression cost, which prevents them from achieving the best balance between accuracy and compression rate. **The joint optimization method proposed in this paper allows for both model accuracy and compression rate to be simultaneously optimized**, delivering superior compression performance.

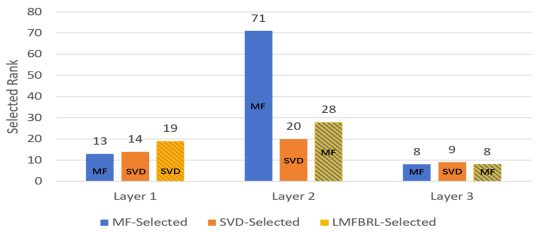

Figure 3: Rank Selection for LeNet-300 Model Layers

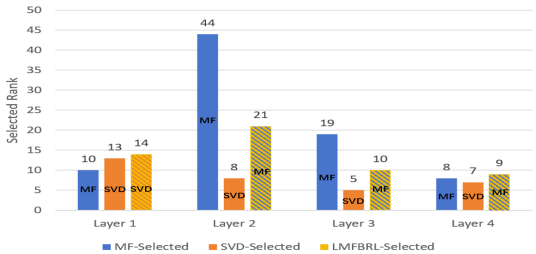

Figure 4: Rank Selection for LeNet-5 Model Layers

Table 4: Model Compression based on Fixed Rank

Model	Method	Params	FLOPs	ACC	ρ_{params}	ρ_{FLOPs}	τ_{params}	τ_{FLOPs}
SVD-Fixed	LeNet-300	35439	141346	97%	86.7%	86.7%	108.4	108.4
	LeNet-5	19973	2640302	81.7%	**95.4%**	**71.2%**	5.7	4.3
	ResNet-20	110852	27798490	85.7%	**59%**	**82.9%**	34.6	48.7
	ResNet-32	178234	45233174	**87.7%**	**61.6%**	**83.6%**	**68.4**	**92.9**
	Vgg-16	453864	199667858	72.5%	97%	84.1%	5.8	5.1
MF-Fixed	LeNet-300	42466	169454	94.8%	84.1%	84.1%	28	28
	LeNet-5	53110	6790650	**97.8%**	87.7%	26%	**175.4**	**52**
	ResNet-20	243463	34376594	84.5%	9.7%	78.8%	3.4	27.1
	ResNet-32	412355	59863312	87.2%	11.2%	78.3%	8	56
	Vgg-16	486393	163030878	69.1%	96.8%	**87%**	4.8	4.3
LMFBRL-Fixed	LeNet-300	34066	135854	**97.2%**	**87.2%**	**87.3%**	**145.3**	**145.3**
	LeNet-5	53110	6790650	**97.8%**	87.7%	26%	**175.4**	**52**
	ResNet-20	186453	35672431	**86.6%**	30.9%	78%	**38.6**	**97.5**
	ResNet-32	213244	51243748	87.6%	54.1%	81.4%	54.1	81.4
	Vgg-16	402326	176310974	**72.8%**	**97.3%**	85.9%	**6**	**5.3**

8.3 Verification of Rank Selection Method in LMFBRL Algorithm

Compared with the rank selected by the fixed decomposition method. This experiment aims to evaluate the effectiveness of the rank selection method proposed in this paper, namely the LMFBRL-Selected algorithm. The algorithms compared in this study are LMFBRL-Selected, SVD-Selected, and MF-Selected, tested on two models: LeNet-300 and LeNet-5.

The LMFBRL-Selected algorithm employs a reinforcement learning-based multi-decomposition method selection framework, which selects the most suitable decomposition method and the optimal rank for each layer of the model. In contrast, SVD-Selected uses a fixed SVD method to compress the model and select the ranks of the decomposed matrices for each layer, while MF-Selected applies the MF method for compression and rank selection.

The results are displayed in Fig. 3, Fig. 4, and Table 3. In the figures, the blue bars represent the ranks chosen by MF-Selected, the orange bars represent the ranks chosen by SVD-Selected, and the LMFBRL-Selected algorithm is illustrated differently: yellow bars with blue stripes indicate that LMFBRL-Selected chose the MF method for that layer, and yellow bars with orange stripes indicate that LMFBRL-Selected chose the SVD method for that layer.

Fig. 3 and Fig. 4 show the rank selections of the three algorithms for the LeNet-300 and LeNet-5 models, respectively. The figures demonstrate that the rank selection for each layer differs across the three algorithms. The LMFBRL-Selected algorithm selects the best decomposition method for each layer, and the ranks chosen by LMFBRL-Selected for the same decomposition method often differ from those selected by the other two algorithms. However, as shown in Table 3, the accuracy of the compressed models for LeNet-300 and LeNet-5 indicates that the LMFBRL-Selected algorithm achieves the highest accuracy. This suggests that the rank selection made by the LMFBRL-Selected algorithm leads to higher accuracy in the compressed models.

These results demonstrate that the rank selection method in the LMFBRL-Selected algorithm is effective. The other two algorithms use fixed decomposition methods and cannot select the optimal decomposition method and corresponding rank for each layer. In contrast, **the LMFBRL-Selected algorithm's rank selection method enables the compressed models to achieve the best accuracy**.

8.4 Verification of Rank Selection Schemes

Compared with the fixed rank decomposition method. This experiment is conducted to demonstrate the effectiveness of the rank selection scheme proposed in this paper. The algorithms compared in this study include LMFBRL-Selected, SVD-Selected, MF-Selected, LMFBRL-Fixed, SVD-Fixed, and MF-Fixed, tested on five models: LeNet-300, LeNet-5, ResNet-20, ResNet-32, and Vgg-16.

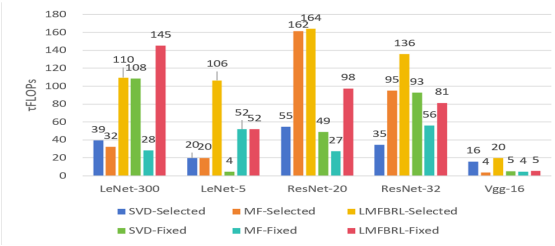

Figure 5: Comparison of τ_{FLOPs} between the Selected-Rank Method and the Fixed-Rank Method

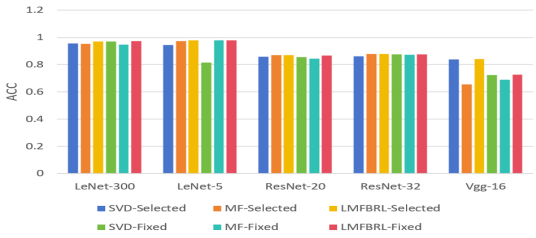

Figure 6: Comparison of ACC between the Selected-Rank Method and the Fixed-Rank Method

The LMFBRL-Selected, SVD-Selected, and MF-Selected algorithms utilize a rank selection scheme, while LMFBRL-Fixed, SVD-Fixed, and MF-Fixed use fixed ranks based on the average rank r obtained from the experiments. The results are presented in Fig.5 and Fig.6, as well as Table 3 and Table 4. Fig.5 and Fig.6 display bar charts comparing the parameters τ_{FLOPs} and ACC, which reflect the compression performance of the algorithms, with specific values detailed in Table 3 and Table 4.

From Fig.5, we observe that the rank selection algorithms in the first three columns generally have higher average τ_{FLOPs} than the fixed rank algorithms in the last three columns, except for the LeNet-300 model, where the rank selection algorithm's average τ_{FLOPs} is slightly lower than that of the fixed rank algorithm. This indicates that the rank selection scheme offers superior compression performance in most models.

Fig.6 shows that all algorithms maintain high accuracy across the models. The average accuracy of the rank selection algorithms for LeNet-5, ResNet-20, and Vgg-16 is higher than that of the fixed rank algorithms, while the other two models show similar average levels between the two types of algorithms. This suggests that the proposed rank selection scheme achieves better accuracy than the fixed rank schemes.

These results confirm the effectiveness of the proposed rank selection scheme, as the three algorithms using fixed ranks cannot adaptively select the optimal rank. In contrast, **the proposed scheme can identify the best rank for each layer of the model**, resulting in improved compression performance.

9 Conclusion

To deploy AI models efficiently on resource-constrained edge devices, model compression techniques are commonly used. This paper focuses on using low-rank decomposition for model compression.

Traditional low-rank decomposition research typically employs a single matrix decomposition method for each parameter matrix in a neural network, neglecting the structural characteristics of each layer and failing to achieve optimal compression. This paper introduces a new scheme called LMFBRL, which uses multiple decomposition methods for model compression, selecting the most suitable method for each layer. However, implementing this plan requires establishing criteria for selecting decomposition methods and balancing model accuracy and compression costs. To address this, we propose a joint optimization approach that simultaneously optimizes model accuracy and compression ratio. We introduce a rank selection scheme to choose the most appropriate rank for each decomposition method and incorporate a reinforcement learning-based framework to select the optimal decomposition method for each network layer.

Experimental results show that the LMFBRL algorithm outperforms other single decomposition methods by selecting the most suitable method for each layer, resulting in higher accuracy and compression rates. The data demonstrate that optimizing both model accuracy and compression cost achieves superior results. Compared to fixed-rank decomposition methods, our rank selection scheme enables the compressed model to achieve higher accuracy and compression rates.

References

Metallinou A Acharya A, Goel R. Online embedding compression for text classification using low rank matrix factorization. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 6196–6203, 2019.

Akamaster. Proper resnet implementation for cifar10/cifar100 in pytorch. `https://github.com/akamaster/pytorch_resnet_cifar10`, 2017.

Feris R S Cheng Y, Yu F X. An exploration of parameter redundancy in deep networks with circulant projections. In *Proceedings of the IEEE international conference on computer vision.*, pages 2857–2865, 2015.

Zhou P Cheng Y, Wang D. Model compression and acceleration for deep neural networks: The principles, progress, and challenges. *IEEE Signal Processing Magazine*, 35(1):126–136, 2018.

Dinh L Denil M, Shakibi B. Predicting parameters in deep learning. *Advances in neural information processing systems*, 26, 2023.

Y. LeCun et al. Lenet-5, convolutional neural networks. http://yann.lecun.com/exdb/lenet, 2015.

Gene H. Golub and Charles F. van Loan. *Matrix computations*. Johns Hopkins University Press, fourth edition, 2012.

Dean J Hinton G, Vinyals O. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

Yang C.-Y Lin H.-Y Lin C.-J Hsu, C.-W. A survey of methods for exploring underlying structures in data of high dimension. In *In Proceedings of the 1st Asian Conference on Machine Learning*, page 134, 2017.

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.

Kyung C M Kim H, Khan M U K. Efficient neural network compression. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 12569–12577, 2019.

Yoo S et al Kim Y D, Park E. Compression of deep convolutional neural networks for fast and low power mobile applications. *arXiv preprint arXiv:1511.06530*, 2015.

Alex Krizhevsky. Cifar-10 and cifar-100 datasets. https://www.cs.toronto.edu/~kriz/cifar.html.

Rakhuba M Lebedev V, Ganin Y. Speeding-up convolutional neural networks using fine-tuned cp-decomposition. *arXiv preprint arXiv:1412.6553*, 2014.

Li W et al Li Y, Adamczewski K. Revisiting random channel pruning for neural network compression. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 191–201, 2022.

Ng M K Liu Y. Deep neural network compression by tucker decomposition with nonlinear response. In *Knowledge-Based Systems*, page 108171, 2022.

Lin W Luo J H, Wu J. Thinet: A filter level pruning method for deep neural network compression. In *Proceedings of the IEEE international conference on computer vision*, pages 5058–5066, 2017.

Qin S Lyu H, Sha N. Advances in neural information processing systems. *Advances in neural information processing systems*, 32, 2019.

Sozykin K et al Phan A H, Sobolev K. Stable low-rank tensor decomposition for compression of convolutional neural network. In *Computer VisionECCV 2020: 16th European Conference, Glasgow, UK, August 2328, 2020*, pages 522–539, 2020a.

Sozykin K et al. Phan A H, Sobolev K. Stable low-rank tensor decomposition for compression of convolutional neural network. In *Computer VisionECCV 2020: 16th European Conference, Glasgow, UK, August 2328, 2020*, pages 522–539, 2020b.

K. Simonyan and A. Zisserman. ery deep convolutional networks for large-scale image recognition. In *In 3rd International Conference onLearning Representations, ICLR 2015, San Diego, CA, USA*, 2015.

Zhang Y Tai C, Xiao T. Convolutional neural networks with low-rank regularization. *arXiv preprint arXiv:1405.3866*, 2015.

Wang J et al Tang J, Chen X. Compressible-composable nerf via rank-residual decomposition. In *Neural Information Processing Systems*, pages 14798–14809, 2022.

Shi X Wang C, Wang H. Automc: Automated model compression based on domain knowledge and progressive search. In *2024 IEEE 40th International Conference on Data Engineering*, 2024.

Madotto A Lin Z Fung PS Winata, G. I. Benchmarking transformer-based model compression on language understanding. *In Proceedings of the 1st Workshop on Energy Efficient Machine Learning and Cognitive Computing for Embedded Applications*, 2015.

Wang Y et al Wu J, Leng C. Quantized convolutional neural networks for mobile devices. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4820–4828, 2016.

Zhang C Wu K, Guo Y. Compressing deep neural networks with sparse matrix factorization. *IEEE transactions on neural networks and learning systems*, 31(10):3828–3838, 2019.

Gong Y et al. Xiao J, Zhang C. Haloc: hardware-aware automatic low-rank compression for compact neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence.*, volume 37, pages 10464–10472, 2023.

Zhang S et al Xu Y, Li Y. Trp: Trained rank pruning for efficient deep neural networks. *arXiv preprint arXiv:2004.14566,*, 2020.

C. Cortes Y. LeCun and C. J. Burges. Mnist handwritten digit database. 2010. http://yann.lecun.com/exdb/mnist, 2010.

Y. Bengio Y. LeCun, L. Bottou and P. Haffner. Gradient-based learning applied to document recognition. In *Proceedings of the IEEE*, volume 86, page 22782324, 1998.

Wen W et al Yang H, Tang M. Learning low-rank deep neural networks via singular vector orthogonality regularization and singular value sparsification. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition workshops*, pages 678–679, 2020.

Cheung N M et al Zhou Y, Moosavi-Dezfooli S M. Adaptive quantization for deep neural network. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.

Mao H et al. Zhu C, Han S. Trained ternary quantization. *arXiv preprint arXiv:1612.01064*, 2016.

A Appendix

A.1 Full-Rank Matrix Factorization (MF)

As show in Fig.7, for the weight matrix $W_i \in \mathbb{R}^{m_i \times n_i}$ of the i-th layer of AI model, the matrix factorization decompose it into the product of matrices $U_i \in \mathbb{R}^{m_i \times r_i}$ and $V_i \in \mathbb{R}^{r_i \times n_i}$, where r_i is the selected rank, and $r_i \leq \min\{m_i, n_i\}$, which can be formulated as follow:

$$W_i \approx U_i \cdot V_i, \tag{12}$$

where the matrix $U_i \in \mathbb{R}^{m_i \times r_i}$ is a row feature matrix, and $V_i \in \mathbb{R}^{r_i \times n_i}$ is a column feature matrix.

We find that the size of the weight matrix W_i is $m_i \times n_i$, while the size of the decomposed matrices U_i and V_i can be denoted as the cost function $C_{MF}(r(W_i))$ as follow:

$$C_{MF}\left(r\left(W_i\right)\right) = r_i \cdot \left(m_i + n_i\right), \tag{13}$$

Therefore, due to the low rank property of the weight matrix W_i, when the chosen rank r_i is sufficiently small, the number of parameters in the decomposed matrix, $r_i \times (m_i + n_i)$, is much less than $m_i \times n_i$. This significantly reduces the number of model parameters, thereby achieving decomposition and compression of the weight matrix in the AI model.

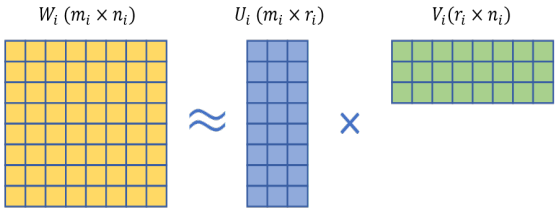

Figure 7: Matirx factorization decomposition diagram.

MF Decomposition for Model Compression Structure. As shown in Fig. 8, the MF method is applied to compress the weight matrix of the i-th layer in the original AI model. The structure of the original model (Fig. 8.a) transforms the weight matrix W_i into the product of two matrices, U_i and V_i, resulting in the compressed model (Fig. 8.b). Thus, the depth of the compressed model increases, while the number of parameters decreases, achieving compression of the original model.

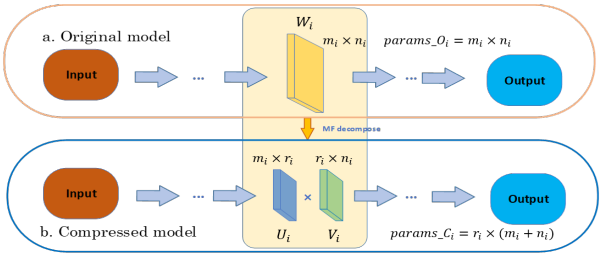

Figure 8: Illustration of the structural changes in model based on MF.

A.2 Singular Value Decomposition (SVD)

As shown in Fig. 9, the weight matrix $W_i \in \mathbb{R}^{m_i \times n_i}$ of the i-th layer in the AI model is factorized into the product of three matrices: $U_i \in \mathbb{R}^{m_i \times r_i}$, $S_i \in \mathbb{R}^{r_i \times r_i}$, and $V_i \in \mathbb{R}^{r_i \times n_i}$. Here, r_i represents the chosen rank, where $r_i \leq \min\{m_i, n_i\}$. This can be expressed as follows:

We observe that the size of the weight matrix W_i is $m_i \times n_i$, and the dimensions of the decomposed matrices U_i, S_i, and V_i can be expressed as the cost function $C_{SVD}(r(W_i))$, as follows:

$$W_i \approx U_i \cdot S_i \cdot V_i^T. \tag{14}$$

where $U_i \in \mathbb{R}^{m_i \times r_i}$ represents the left singular matrix, $V_i \in \mathbb{R}^{n_i \times r_i}$ represents the right singular matrix, and $S_i \in \mathbb{R}^{r_i \times r_i}$ represents the singular value matrix

$$C_{SVD}\left(r\left(W_i\right)\right) = r_i \cdot \left(m_i + r_i + n_i\right). \tag{15}$$

Therefore, due to the low rank property of the weight matrix W_i, when the chosen rank r_i is sufficiently small, the number of parameters in the decomposed matrices, $r_i \times (m_i + r_i + n_i)$, is significantly less than $m_i \times n_i$. This reduction greatly decreases the number of model parameters, enabling effective decomposition and compression of the weight matrix in the AI model.

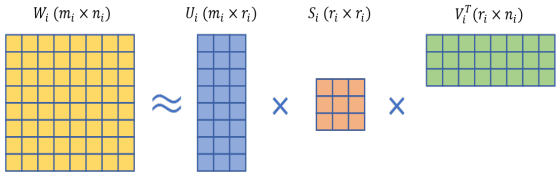

Figure 9: Singular Value Decomposition diagram.

SVD for Model Compression Structure. As illustrated in Fig. 10, the SVD method is employed to compress the weight matrix of the i-th layer in the original AI model. The original model's structure (Fig. 10.a) transforms the weight matrix W_i into the product of three matrices: U_i, S_i, and V_i, resulting in the compressed model (Fig. 10.b). Consequently, the depth of the compressed model increases while the number of parameters decreases, effectively compressing the original model.

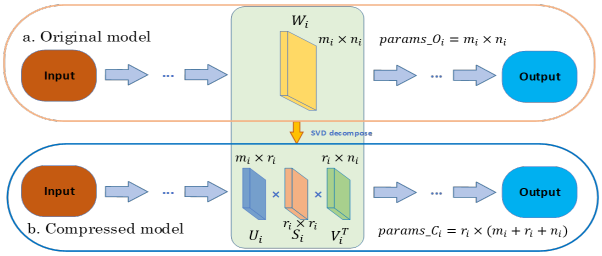

Figure 10: Illustration of the structural changes in model based on SVD.

A.3 Reinforcement Learning Introduction

Reinforcement learning adjusts each step of action based on feedback from the current environment, ultimately maximizing overall rewards. In this paper, the current environment for the reinforcement learning framework refers to the i-th layer of the model, where applying different decomposition methods to the weight matrix yields different feedback. The goal is to select the decomposition method with the highest feedback for the current layer, maximizing the overall reward. This paper employs the Q-learning algorithm based on a greedy strategy. According to the following Eq. (16):

$$K^*(h_i, b_i) = \mathbb{Q}\left[Q_i + \epsilon \cdot \max_b K^*(h_{i-1}, b_{i-1})\right]. \tag{16}$$

Among them, $h_i \in H$ is the state subspace, which is the set of decomposition methods chosen for each network layer. $b_i \in B$ is the action subspace, representing our comparison operation for each layer's decomposition method. $K^*(*)$ represents the optimal action value function, which represents the overall benefit of selecting a decomposition method for the compressed model. Q_i represents the maximum reward of the current state, indicating the benefit after selecting the decomposition method for the i-th network layer. $\mathbb{Q}$ represents the maximum expected overall reward, and ϵ is a hyperparameter.

As shown in Fig. 11, the entire model is in state space H, and the decomposition method chosen for each network layer in the model is action space B. The Q in Fig. 11 represents the reward value obtained for the corresponding network layer after implementing the two decomposition methods. We need to compare the value Q obtained from each decomposition method, select the decomposition method with the highest value based on the greedy strategy, and finally maximize the profit of the entire model. In the figure 11, by comparing the value Q of SVD and MF decomposition methods at each network layer, the selected method is marked with blue fluorescence.

Finally, the entire chain of blue fluorescence is our optimal selection scheme, which can achieve the best decomposition compression effect of the model.

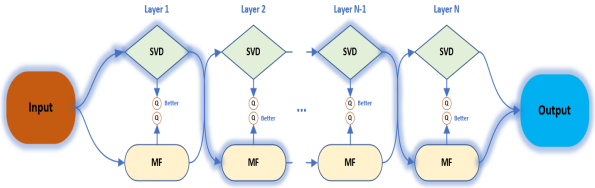

Figure 11: Selections of multi-decomposition methods based on reinforcement learning.

A.4 Details Of Experience

We test the algorithm proposed in this paper using the following five models:

- **LeNet-300(Y. LeCun and Haffner, 1998)**: The LeNet-300 model consists of three fully connected layers, with 300, 100, and 10 neurons in each layer respectively. Sigmoid activation functions are used between each fully connected layer, and a Dropout layer is employed between the first and second fully connected layers to reduce overfitting.
- **LeNet-5(et al., 2015)**: LeNet-5 is a classic convolutional neural network model that consists of two convolutional layers, two pooling layers, and three fully connected layers. The convolutional layers have 5x5 kernels, and the fully connected layers contain 120, 84, and 10 neurons, respectively.
- **Resnet-20(Akamaster, 2017)**: ResNet-20 is a basic version of the Residual Network (ResNet), consisting of 20 convolutional layers. Its structure includes several convolutional layers, batch normalization layers, global average pooling layers, and a fully connected layer for classification.
- **Resnt-32(Zhu C, 2016)**: Compared to ResNet-20, ResNet-32 has a deeper network architecture, consisting of 32 convolutional layers. The model structure includes a series of convolutional layers, batch normalization layers, and global average pooling layers, followed by a fully connected layer for classification.
- **Vgg-16(Simonyan and Zisserman, 2015)**: The VGG16 model consists of 16 layers, including 13 convolutional layers and 3 fully connected layers. This model utilizes small 3x3 convolutional kernels and a deep network structure, with multiple convolutional layers stacked repeatedly.

We conducted experimental tests on the aforementioned model using six low-rank decomposition compression algorithms:

- **SVD-Selected**: This method employs SVD (see Section A.2), where the original matrix is decomposed into the product of the left singular vectors, a diagonal matrix of singular values, and the transpose of the right singular vectors. We use the SVD adaptive rank selection algorithm from Yerlan et al.'s (Idelbayev and Carreira-Perpinán, 2020) research for testing and comparison.
- **SVD-Fixed**: This method also uses SVD, but the rank of the decomposed matrix is fixed. The rank is determined by averaging the ranks obtained from the experiments conducted in this study.
- **MF-Selected**: This method uses MF decomposition (see Section A.1), where the original matrix is decomposed into the product of two low-rank matrices. We derived a corresponding MF adaptive rank selection algorithm by replacing the SVD in Yerlan et al.'s adaptive rank selection algorithm with the MF decomposition method.
- **MF-Fixed**: This method also uses MF decomposition, but the rank of the decomposition matrices is fixed. The rank is determined by averaging the ranks obtained from the experiments conducted in this study.
- **LMFBRL-Selected**: This paper proposes the LMFBRL-Selected algorithm, which is based on a reinforcement learning framework. This algorithm selects the most suitable decomposition method for each network layer to achieve optimal model compression performance. The decomposition methods adaptively select the rank.
- **LMFBRL-Fixed**: This paper proposes the LMFBRL-Fixed algorithm, which follows the same framework as LMFBRL-Selected. However, in LMFBRL-Fixed, the decomposition methods use fixed ranks.

We trained and tested the above-mentioned models using two different datasets:

- **MNIST(Y. LeCun and Burges, 2010)**: The MNIST dataset consists of hand-written digit images from 250 different people, covering the digits 0 through 9. Each image is a grayscale image with a size of 28×28 pixels. The training set contains 60,000 images, while the test set contains 10,000 images.
- **CIFAR-10(Krizhevsky)**: The CIFAR-10 dataset consists of 60,000 color images divided into 10 classes, with each class containing 6,000 images. The images in the dataset are 32x32 pixels in size. The training set comprises 50,000 images, while the test set contains 10,000 images.

We use seven metrics to evaluate the performance of the proposed algorithm:

- **Params**: The total number of parameters that need to be trained in the model.
- **FLOPs**: The number of floating-point operations required for a single forward pass through the model.
- **Accuracy (ACC)**: The accuracy of the model on the test set, defined as the proportion of correctly classified samples to the total number of samples.
- $\rho_{\textbf{params}}$: The percentage reduction in the number of parameters after compression relative to the original model. It is calculated as:
$$\rho_{\text{params}} = \left(1 - \frac{\text{Params(compressed)}}{\text{Params(original)}}\right) \times 100\%. \tag{17}$$
- $\rho_{\textbf{FLOPs}}$: The percentage reduction in the number of floating-point operations after compression relative to the original model. It is calculated as:
$$\rho_{\text{FLOPs}} = \left(1 - \frac{\text{FLOPs(compressed)}}{\text{FLOPs(original)}}\right) \times 100\%. \tag{18}$$
- $\tau_{\textbf{params}}$: The parameter compression ratio per unit of accuracy loss, reflecting the effectiveness of parameter compression. It is calculated as:
$$\tau_{\text{params}} = \frac{\rho_{\text{params}}}{\text{ACC(original)} - \text{ACC(compressed)}}. \tag{19}$$
A higher value of τ_{params} indicates a better compression effect in params.
- $\tau_{\textbf{FLOPs}}$: The computation compression ratio per unit of accuracy loss, reflecting the effectiveness of computation compression. It is calculated as:
$$\tau_{\text{FLOPs}} = \frac{\rho_{\text{FLOPs}}}{\text{ACC(original)} - \text{ACC(compressed)}}. \tag{20}$$
A higher value of τ_{FLOPs} indicates a better compression effect in FLOPs.

Our experimental setup includes an NVIDIA GeForce RTX 4070 GPU, 32GB of RAM, running in Windows 11. We used Python version 3.7 and PyTorch version 1.31.0. The experiments were conducted on 5 AI models using 2 datasets and 5 algorithms.