IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received 20 June 2024, accepted 8 July 2024, date of publication 15 July 2024, date of current version 23 July 2024.

Digital Object Identifier 10.1109/ACCESS.2024.3428370

RESEARCH ARTICLE

Rank Selection Method of CP Decomposition Based on Deep Deterministic Policy Gradient Algorithm

SHAOSHUANG ZHANG, ZHAO LI, WENLONG LIU, JIAQI ZHAO, AND TING QIN
School of Computer Science and Technology, Shandong University of Technology, Zibo 255049, China

Corresponding author: Zhao Li (lizhao_buaa@126.com)

This work was supported in part by the National Key Research and Development Program of China under Grant 2022YFE0107300, and in part by the Key Research and Development Program of Shandong Province (Soft Science Project) under Grant 2023RKY01010.

ABSTRACT With the popularity of edge computing devices and increasing complexity of convolutional neural network (CNN) models, the need for model compression and acceleration has become increasingly urgent. As an effective model compression technique, CANDECOMP/PARAFAC (CP) decomposition relies heavily on the preset rank for its compression effectiveness. However, no direct algorithm is currently available for determining the optimal tensor rank. Therefore, a novel method for CP rank selection based on deep reinforcement learning is proposed. This method utilizes the DecG single-player game framework based on the Deep Deterministic Policy Gradient (DDPG) algorithm to achieve automation and intelligence in rank selection. In this process, a pre-trained model is introduced, which fuses and reshapes several historical tensors as network inputs. Additionally, a hybrid greedy strategy based on singular value decomposition (SVD) was designed in the exploration phase to enhance the efficiency of finding ideal rank selection results. This method can automatically determine the rank according to the weight tensor characteristics of the convolution layer and optimize the compression efficiency and performance of the model. In addition, a compression efficiency index is developed to visually demonstrate the performance of the various compression methods. Finally, on the CIFAR-10 and CIFAR-100 datasets, CP decomposition experiments are conducted on various convolutional neural network models, and the decomposed models undergo iterative fine-tuning for retraining. The experimental results show that the rank values determined by the DecG method achieve significant optimization and enhancement in the compression efficiency of the models compared to other methods, exhibiting strong robustness.

INDEX TERMS Convolutional neural network, deep deterministic policy gradient, model compression, low-rank decomposition, CP rank selection.

I. INTRODUCTION

The development of CNNs has shown excellent performance in solving computer vision problems such as image classification and recognition [1], [2], [3], and the number of parameters has increased from approximately 60,000 in the early LeNet-5 network [4] to hundreds of millions. For example, the VGG19 model has been widely used owing to its excellent performance in various computer vision tasks, and the number of parameters reaches approximately 144 million [5]. The number of parameters has become extremely large, leading to the consumption of a significant amount of time and space during training and inference processes [6]. The use of edge computing platforms tends to integrate advanced technologies, such as artificial intelligence, to enhance real-time data processing and decision-making at the edge of the network. However, as edge devices often have limited resources, reducing the model size and computational complexity while ensuring model performance has become a critical issue for deploying

The associate editor coordinating the review of this manuscript and approving it for publication was Xianzhi Wang.

large neural networks on edge computing platforms [7]. Low-rank decomposition can not only reduce the storage requirements for parameters but also decrease the amount of computation by decomposing high-dimensional parameter vectors into sparse low-dimensional vectors. Model compression and acceleration can be achieved using the low-rank decomposition [8].

Low-rank decomposition methods can be roughly divided into three categories: two-component decomposition, three-component decomposition, and four-component decomposition [9]. The core difference among these three decomposition methods lies in the number of decompositions applied to the weight tensor. Specifically, as the number of decompositions of the weight tensor increases, it can reveal the intrinsic structure and characteristics of the data more comprehensively. Therefore, for model compression, the CP decomposition, a commonly used four-component tensor decomposition method in mathematics, can be used [10], in comparison with other low-rank decomposition methods, the advantage of CP decomposition is that the four matrices obtained from the fourth-order tensor decomposition have the same latent variable dimension, which can reduce the number of parameters that need to be adjusted. In addition, CP decomposition exhibits a certain degree of robustness to noise and outliers. In some cases, even in the presence of noise or outliers, CP decomposition can still achieve good decomposition results and model performance. Therefore, in this study, CP decomposition was selected as the method for compressing the model.

However, solving the tensor rank is an NP-hard problem [11]. Therefore, in CP decomposition, the desired tensor rank is a predefined parameter, implying that it should be specified before decomposition. If the rank is set too high, overfitting and increased computational complexity may occur. If the rank is set too low, the data structure in the tensor may not be captured adequately. Currently, there are numerous CP rank selection methods aimed at overcoming the challenge of manually and time-consumingly specifying the CP rank. However, they are suboptimal solutions and can easily fall into local optimal solutions in some specific situations, exhibiting poor robustness.

To address the issue of the poor robustness of existing CP rank selection methods, this study proposes a DecG framework based on DDPG to obtain a CP rank selection strategy that can achieve better compression effects for CNN models and reduce resource costs. This framework employs a hybrid greedy strategy based on singular value decomposition and redefines the environmental state using it as the network input information. Ingeniously converting the complex rank selection problem into a single-player game, it utilizes a self-constructed training set to pre-train the agent's network model and leverages advanced optimization algorithms to automatically explore the rank with the highest compression benefit in each convolutional layer.

The remainder of this paper is organized as follows. In Section II, the low rank decomposition and classical CP rank selection methods are reviewed. In Section III, we introduce a CP rank selection method based on the DecG framework. The evaluation criteria and experimental results obtained using different rank estimation methods are presented in Section IV. Finally, Section V summarizes the study and presents future challenges.

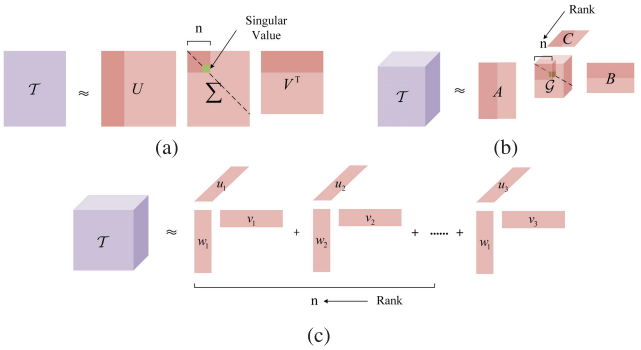


FIGURE 1. Diagram of three decomposition methods. (a)singular value decomposition. (b)Tucker decomposition in the third-order tensor case. (c)CP decomposition in the third-order tensor case.

II. RELATED WORK

Among the low-rank decomposition methods, singular value decomposition (SVD), Tucker decomposition (TD), and CP decomposition are three typical methods (see Fig. 1).

SVD can decompose any matrix into a singular value matrix and two characteristic matrices, as shown in the formula below:

$$\mathcal{T} = U \Sigma V^{\mathrm{T}} \tag{1}$$

where Σ is a singular value matrix, which is a diagonal matrix. Matrices U and V represent the eigenvectors of matrix $\mathcal{T}$ in different directions. The magnitude of the singular values reflects the importance of the eigenvectors in the corresponding directions, which means that the matrix can be approximately described by the first n singular values with larger magnitudes and their corresponding eigenvectors.

Initially, Denil et al. [12] proposed a two-component decomposition method based on SVD to compress models. However, the size of one of the tensor dimensions after two-component decomposition is still relatively large, which limits the improvement in compression effects to a certain extent. To solve this problem, Denton et al. [13] proposed the use of SVD for three-component decomposition to reconstruct the weights of the fully connected layer. With further research, more improved methods based on SVD [14], [15], [16] have emerged, which have been successfully applied to the compression of the entire CNN.

Meanwhile, Kim et al. [17] proposed the use of Tucker decomposition to compress all the convolutional layers and fully connected layers, further decomposing one of the tensors after the two-component decomposition to obtain convolutions of $w \times 1$, $1 \times h$, and 1×1. Tucker decomposition is a high-order version of Principal Component Analysis (PCA), which decomposes a tensor into the product of a core

tensor and corresponding matrices in each dimension. Taking a third-order tensor $\mathcal{T} \in \mathbb{R}^{I\times J\times K}$ as an example, the Tucker decomposition is expressed as follows:

$$\mathcal{T} \approx \mathcal{G}\times_1 A\times_2 B\times_3 C \tag{2}$$

where the symbol $\times_k, k = 1, 2, 3$ represents the product of a tensor and a matrix, which is also known as the modal product. As can be seen from the formula, the Tucker decomposition is not limited to the dimensions of the tensor and can be used as a two-component decomposition, three-component decomposition [18], or four-component decomposition [19].

CP decomposition is a special case of the Tucker decomposition [20], which is usually a four-component decomposition. The CP decomposition process involves decomposing a higher-order tensor into the sum of multiple rank-one tensors. For example, the third-order tensor data $\mathcal{T} \in \mathbb{R}^{I\times J\times K}$ can be decomposed into:

$$\mathcal{T} \approx \sum_{r=1}^{R} u_r \circ v_r \circ w_r \tag{3}$$

where $\circ$ represents the outer product of the tensor, $u_r \in \mathbb{R}^I$, $v_r \in \mathbb{R}^J$, and $w_r \in \mathbb{R}^K$ are all vectors, called factor vectors of CP decomposition, when $r = 1, \ldots, R$, u_r, v_r, and w_r can respectively form the corresponding factor matrices $U \in \mathbb{R}^{I\times R}$, $V \in \mathbb{R}^{J\times R}$, and $W \in \mathbb{R}^{K\times R}$, so it can also be written:

$$\mathcal{T}_{ijk} \approx \sum_{r=1}^{R} U_{ir} V_{jr} W_{kr}$$
$$for\ i = 1, \ldots, I, j = 1, \ldots, J, k = 1, \ldots, K \tag{4}$$

Furthermore, for the third-order tensor $\mathcal{T}$, in low-rank tensor approximation, the minimum rank of the tensor data is obtained by minimizing data fitting, and the approximation problem is described as an optimization problem as follows:

$$\min_R \frac{1}{2}\left\|\mathcal{T} - \sum_{r=1}^{R} u_r \circ v_r \circ w_r\right\|_F^2 \tag{5}$$

where $\|\cdot\|_F^2$ denotes the Frobenius norm.

In CNNs, $\mathcal{O} \in \mathbb{R}^{X\times Y\times S}$ is used as the input tensor and mapped to the output tensor $\mathcal{P} \in \mathbb{R}^{(X-d+1)\times(Y-d+1)\times T}$, which is formulated as follows:

$$\mathcal{P}(x, y, t) = \sum_{i=x-\delta}^{x+\delta}\sum_{j=y-\delta}^{y+\delta}\sum_{s=1}^{S} \mathcal{A}(i-x+\delta, j-y+\delta, s, t)\mathcal{O}(i, j, s) \tag{6}$$

where δ is $(d-1)/2$, and the four-dimensional tensor $\mathcal{A} \in \mathbb{R}^{d\times d\times S\times T}$ represents the convolutional weights. $d \times d$ corresponds to the dimensions of the kernel space, S corresponds to the different input channels, and T corresponds to the different output channels. The rank-R CP decomposition of tensor $\mathcal{A}$ can be derived from (4) as follows:

$$\mathcal{A}(i, j, s, t) = \sum_{r=1}^{R} A^x(i-x+\delta, r)\, A^y(i-y+\delta, r)\, A^s(s, r)\, A^t(t, r) \tag{7}$$

where $A^x \in \mathbb{R}^{d\times R}$, $A^y \in \mathbb{R}^{d\times R}$, $A^s \in \mathbb{R}^{S\times R}$, and $A^t \in \mathbb{R}^{T\times R}$ are the factor matrices. By substituting (7) into (6), the convolution approximate evaluation expression in (6) is obtained as follows:

$$\mathcal{P}(x, y, t) = \sum_{r=1}^{R} A^t(t, r)\left(\sum_{i=x-\delta}^{x+\delta} A^x(i-x+\delta, r)\left(\sum_{j=y-\delta}^{y+\delta} A^y(j-y+\delta, r)\left(\sum_{s=1}^{S} A^s(s, r)\,\mathcal{O}(i, j, s)\right)\right)\right) \tag{8}$$

Therefore, input tensor $\mathcal{O}$ can be transformed into output tensor $\mathcal{P}$ through a sequence of four convolutions with smaller kernels:

$$\begin{aligned}
\mathcal{O}^s(i, j, s) &= \sum_{s=1}^{S} A^s(s, r)\,\mathcal{O}(i, j, s)\\
\mathcal{O}^{sy}(i, y, r) &= \sum_{j=y-\delta}^{y+\delta} A^y(j-y+\delta, r)\,\mathcal{O}^s(i, j, r)\\
\mathcal{O}^{syx}(x, y, r) &= \sum_{i=x-\delta}^{x+\delta} A^x(i-x+\delta, r)\,\mathcal{O}^{sy}(i, y, r)\\
\mathcal{P}(x, y, t) &= \sum_{r=1}^{R} A^t(t, r)\,\mathcal{O}^{syx}(x, y, r)
\end{aligned} \tag{9}$$

Therefore, the original convolutional layer can be approximated using four decomposition layers. For CNN convolutional layers, CP decomposition is a highly effective compression method [21], [22].

Regarding the solution to (5), unlike the matrix rank, the currently known deterministic methods can only determine the upper bound for the tensor rank. For example, for a general third-order tensor $\mathcal{X} \in \mathbb{R}^{I\times J\times K}$, rank follows the following constraint:

$$\text{rank}(\mathcal{X}) \le \min\{IJ, IK, JK\} \tag{10}$$

Currently, more people are using mathematical and artificial intelligence methods to select a CP rank with better robustness. Fully Bayesian CP Factorization (FBCP) [23] developed an efficient deterministic Bayesian inference algorithm and constructed a hierarchical probabilistic model for predicting the tensor rank. Bayesian Robust Tensor Factorization (BRTF) [24] adopts a fully Bayesian generative model for automatic CP rank estimation. Variational Bayesian Matrix Factorization (VBMF) [25] decomposes the original matrix into the product of a series of submatrices, thereby reducing the rank of the original matrix. By optimizing the parameters in this decomposition process,

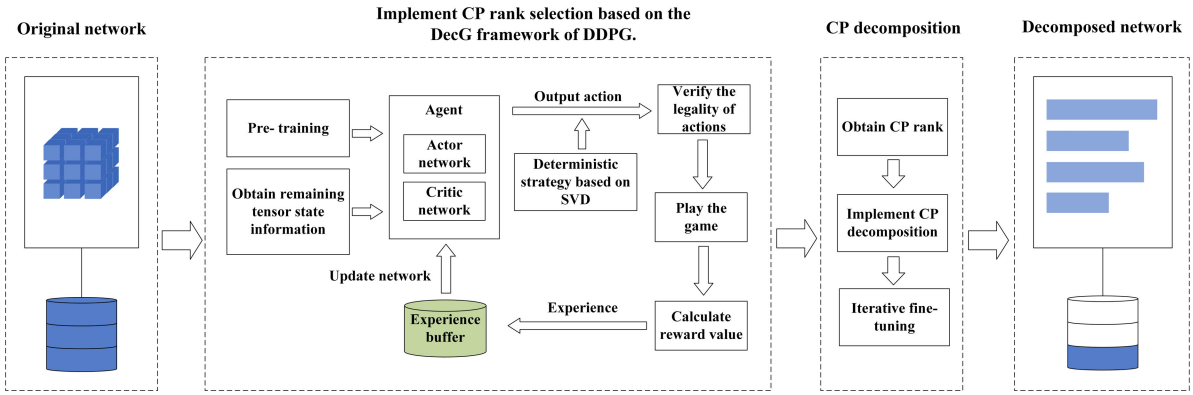


FIGURE 2. Illustration of the process of CP decomposition for CNN models.

the VBMF algorithm can gradually approximate the optimal rank of the original matrix, and can be used to solve the CP rank problem. However, current methods still face challenges such as insufficient robustness, which, to some extent, limits the maximization of model compression benefits.

To further determine the optimal rank and achieve a more efficient model compression performance, Zhou et al. [26] proposed a method for estimating the CP tensor rank from noisy measurements based on a CNN by adding pre-decomposition for feature acquisition and inputting rank-one components into the CNN. This method requires a predefined rank bound to enable the convolutional neural network to extract features effectively, thereby achieving accurate prediction of CP rank. The rank bound used for pre-decomposition must still be manually selected. Because the rich space of matrix multiplication can be formalized as a low-rank decomposition of a specific three-dimensional (3D) tensor for optimization and adjustment, Fawzi et al. [27] used Deep Reinforcement Learning (DRL) to identify and analyze patterns and features in tensor data and used learned agents to predict effective decompositions, which can be extended to deal with related original mathematical problems, such as solving the CP rank. Combining the above ideas, this study proposed a new method for CP rank selection based on the DDPG algorithm and designs a DecG single-player game framework to achieve automation and intelligence in rank selection.

III. CP RANK LEARNING METHOD

To compress models by applying CP decomposition to CNN convolutional layers, DRL can be used to search for the correct and effective CP ranks. DRL can handle various complex combinatorial problems and has strong generalization capabilities [28]. The DDPG algorithm [29] was designed for environments with continuous action spaces, making it suitable for the precise adjustment required for CP rank selection. Therefore, this study designed a single-player game framework called DecompositionGame (DecG) based on the DDPG algorithm for CP rank selection. Subsequently, the obtained CP ranks were used for model compression (see Fig. 2).

During each round of the game loop, the remaining tensor information is first captured and processed precisely. Subsequently, the processed tensor is used as the input state C for the neural network (state C represents the current environmental situation faced by the agent). This state is then passed to a pre-trained agent for computation to obtain the corresponding output. Next, the output information is carefully processed and validated to ensure compliance with the rules of the game mechanism, thereby generating the game action a (action a is the decision made by the agent based on the current state C and the learned strategy). Action a is executed in the game environment, and the corresponding reward values are awarded based on the execution results. These reward values are stored as experiences in an experience buffer and utilized to update and optimize the neural network.

As the game proceeded, the remaining tensors are gradually decomposed until the end of the game. Through this process, the CP ranks required for the CNN model can be calculated accurately. Subsequently, the obtained CP ranks are applied to CP decomposition of the CNN model. After the decomposition is completed, iterative fine-tuning techniques are utilized to restore and optimize network performance.

A. STATE ACQUISITION

In the process of obtaining the decomposition factor vector of the tensors and obtaining the CP rank using agents in the DecG framework, the shape of the fourth-order convolutional weight $\mathcal{Q} \in \mathbb{R}^{I\times J\times K\times L}$ to be decomposed is first used as the shape of the original tensor $\mathcal{M}_0$, $\mathcal{M}_0$ is a randomly generated tensor based on the shape of the convolutional weight that will be decomposed. As the game progresses, the remaining tensor after step θ is $\mathcal{M}_\theta$, and the process can be derived from (3) as follows:

$$\begin{aligned} \mathcal{M}_\theta &= \mathcal{M}_{\theta-1} - u_\theta \circ v_\theta \circ w_\theta \circ x_\theta \\ \theta &= 1, \ldots, R \end{aligned} \tag{11}$$

where R represents the rank of the tensor $\mathcal{M}_0$. Therefore, when $\theta = R$, $\mathcal{M}_R = 0$, the DecG process ends, yielding $\mathcal{M}_0 = \sum_{\theta=1}^{R} u_\theta \circ v_\theta \circ w_\theta \circ x_\theta$, Each set of vectors

Algorithm 1 Get the State C

Require: the remaining tensor $\mathcal{M}_\theta$
Ensure: the state C
1: frame = Reshape ($\mathcal{M}_\theta$)
2: **if** the current remaining tensor is $\mathcal{M}_0$ **then**
3: frames = [frame] * 4 // List copy operation
4: **else**
5: frames.append (frame)
6: frames.pop (0) // Perform an operation to remove the first element from frames
7: **end if**
8: C = expand_dims (frames) //Add one dimension to the zeroth dimension
9: **return** the state C

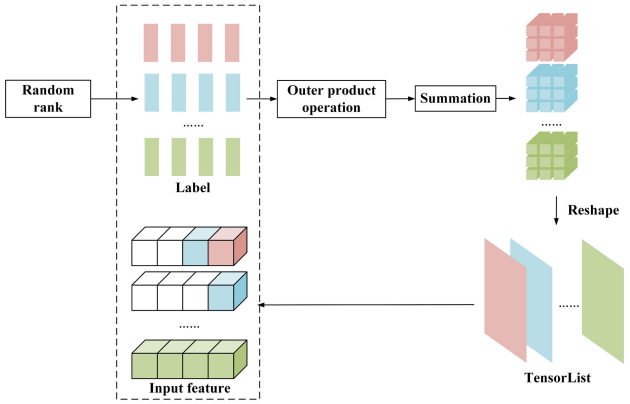


FIGURE 3. Diagram of dataset generation process.

$(u_\theta, v_\theta, w_\theta, x_\theta)_{\theta=1,\ldots R}$ represents an action a in each step of the DecG.

In the DecG process, the remaining tensor $\mathcal{M}_\theta$ is not directly used as input data for the neural network. To further enhance the prediction accuracy and generalization capability of the model for the residual network, we discarded the method of directly utilizing a single historical remaining tensor as the state C for input into the neural network for prediction purposes. Instead, we adopt a reshaping operation to fuse the four historical remaining tensors, creating a tight association among them. This provides more comprehensive and accurate information support for subsequent prediction, increasing the generalization and smoothness of the neural network output, as described in Algorithm 1.

In Algorithm 1, a dimensionality reduction operation is first performed to reduce the dimensions of the current remaining tensor from four to two. Subsequently, a judgment is made to determine whether the current tensor is the original tensor $\mathcal{M}_0$. If it is determined to be the original tensor, a list repetition operation is applied to the reduced-dimension tensor to expand its dimensions to three. Next, whenever the remaining tensor changes, a stack operation is performed to update the data in real-time and ensure that the zeroth dimension of the current tensor always maintains four elements. Finally, to maintain consistency with the input shape of the neural network, an additional dimension is added to ensure the accuracy of the data format and to prepare it for subsequent neural network processing. This approach can enhance the data quality and improve the training effects of the model.

B. PRE-TRAINING

The processed data from Section III-A are inputted as state C into the neural network. The DDPG algorithm updates the network through deterministic policies by utilizing an actor-critic approach. In this approach, the actor policy network $\mu\,(\mathrm{C}|\vartheta^\mu)$ controls the agent to obtain the next action a based on the state C, whereas the critic value network $Q\left(\mathrm{C}, a|\vartheta^Q\right)$ does not directly control the agent. Instead, it evaluates the value of action a after the agent obtains it based on state C. ϑ^μ and ϑ^Q represent the weights of these two networks. Before officially starting the game, we conducted pre-training operations on the actor network to enhance its performance and overcome the problem of insufficient data.

The environments encountered by reinforcement learning are often unknown and dynamic, and typically require a large amount of sample data to train the model. Pre-trained models can be trained on large-scale labeled data, thereby improving the generalization ability and efficiency of the model. Pre-trained models have potential advantages in reinforcement learning [30], [31], [32].

Therefore, to enable the algorithm to quickly identify the optimization direction during the initial stage of training, reduce the risk of falling into local optima, and thereby improve the convergence speed and performance, in this study, we have curated 300,000 sets of data for each unique weight tensor size, constituting a pre-training dataset. Consistent with other experimental methodologies, this dataset was processed utilizing the AMD EPYC 9754 CPU in conjunction with the IDIA Tesla T4 GPU. With a total batch size set at 1,024, 1,000 iterations were configured for the actor network within the agent, enabling the network model to promptly discern the optimal direction during the training procedure and accelerate the convergence process (see Fig. 3).

To construct the dataset, this study followed the derivation steps outlined in (11), gradually obtaining $\mathcal{M}_0$ from $\mathcal{M}_\theta$. First, the CP rank value for each dataset is determined. To ensure data diversity, a randomization method is employed to generate random vectors based on the desired shapes of the factor vectors. Subsequently, these random vectors are used to obtain the corresponding tensors through the outer product operation. The remaining tensor for the current iteration was derived by adding the currently obtained tensor to the previous tensor. All the remaining tensors are then subjected to dimensionality reduction and stored in a list in a two-dimensional format. Subsequently, starting from the last data item in the list, stack operations are gradually performed forward according to Algorithm 1, constructing four-dimensional tensor input features. These input features,

Algorithm 2 Mixed Greed Strategy

Require: exploration factors delta1 and delta2, state C
Ensure: the action a

```
epsilon1 = [1., .1]
epsilon2 = [1., .1]
while not game_over do
    if np.random.random() < epsilon1[0] then
        if np.random.random() < epsilon2[0] then
            a = Random(C)
        else
            a = SVD(C)
        end if
        Epsilon2[0]-=delta2
    else
        a = model.predict(C)
    end if
    epsilon1[0]-=delta1
end while
return the action a
```

along with random vectors serving as labels, collectively comprise the complete dataset.

C. HYBRID GREEDY STRATEGY

To obtain the action a, we improve the action selection strategy in the DDPG algorithm by combining the original random action with the deterministic strategy based on SVD and implementing a hybrid greedy strategy to enhance the convergence speed and performance of the algorithm.

In the traditional DDPG algorithm, the action selection is initially random. As the training progresses, the algorithm gradually trusts the action output of the network and increases their frequency of use. However, this study introduces a combination of a deterministic strategy based on SVD and a random strategy as an exploration strategy in the exploitation and exploration processes. The SVD process can be derived from (1), where the magnitude of the singular values reflects the importance of the corresponding feature vectors in each direction. From this, the maximum factor matrix can be obtained and used as an exploration strategy, as follows:

$$N = max\left\{\mathcal{T}\right\} \tag{12}$$

By incorporating this exploration strategy into the actor policy, the resulting strategy is obtained as follows:

$$\mu'\left(\mathrm{C}\right) = \mu\left(\mathrm{C}|\vartheta^{\mu}\right) + \zeta + N \tag{13}$$

where ζ represents the random strategy, as detailed in Algorithm 2.

As a powerful matrix decomposition technique, SVD can extract key features from data. Here, SVD is used to analyze historical action data, thereby obtaining a deterministic strategy that enables the algorithm to select actions based on the intrinsic structure of the data, rather than relying solely on randomness or network outputs. This strategy retains randomness during the initial stages of training to explore more possibilities, gradually relies on the results of SVD analysis as training progresses, and finally trusts the network, resulting in more precise and efficient action selection.

After obtaining results based on the exploitation and exploration strategies, they are subjected to validity verification to ensure that the actual output state is consistent with the expected output. If the verification is successful, the result is placed in the game environment for further learning. If the verification fails, an exception is immediately discarded to facilitate quick problem identification, ensuring the rigor and accuracy of the learning process.

Next, the game session is initiated, and based on the data information of state C, it is accurately transformed into the remaining tensor $\mathcal{M}_{\theta-1}$. Subsequently, this tensor and the action a are substituted into (11) to obtain a new remaining tensor $\mathcal{M}_{\theta}$. For the remaining tensor $\mathcal{M}_{\theta}$, the score g after executing the action is calculated according to the established reward mechanism. To ensure the efficiency and rationality of the game process, a limit is placed on the maximum number of steps θ to prevent the game from proceeding indefinitely because of the excessive pursuit of penalty reduction. Therefore, after each increment in the number of steps, the reward and penalty mechanisms will impose an additional penalty of $-\varphi\left(\theta\right)$, where $\varphi\left(\theta\right)$ increases with the increase in θ. In the case of successful decomposition, a conditional incremental update is applied to the final ten steps, introducing a decay rate to reduce the magnitude of the increase. This value is calculated only upon the conclusion of the current game, thereby incentivizing the game to minimize the number of steps and select the CP rank when choosing favorable actions. In addition, a limit $\{\min, \max\}$ is set, and if the range of elements in the remaining tensor $\mathcal{M}_{\theta}$ exceeds this limit, the game will immediately terminate and be judged as a failure, thus avoiding an infinite loop caused by continuously performing unfavorable actions.

Finally, the round experience is stored in a buffer, and the network is trained through random sampling. Based on the previous operations, the next state C' can be calculated from the obtained remaining tensor $\mathcal{M}_{\theta}$, and the set $\left(\mathrm{C}, a, \mathrm{g}, \mathrm{C}', \mathrm{f}\right)$ is stored as memory. When required, samples are collected to obtain the actual target value as follows:

$$q\left(\mathrm{g}, \mathrm{C}', \mathrm{f}\right) = \mathrm{g} + \gamma\left(1 - \mathrm{f}\right) Q'\left(\mathrm{C}', \mu'\left(\mathrm{C}'|\vartheta^{\mu'}\right)|\vartheta^{Q'}\right) \tag{14}$$

where γ represents the discount factor, f represents the progress of the game (with 1 indicating success and 0 indicating failure), Q' and μ' correspond to the target networks of the critic value network and actor policy network, respectively. The critic value network is updated by performing gradient descent on the loss function *Loss* with respect to the parameter ϑ, as shown in (15):

$$Loss = \frac{1}{|B|}\sum\left(q\left(\mathrm{g}, \mathrm{C}', \mathrm{f}\right) - Q\left(\mathrm{C}, a|\vartheta^{Q}\right)\right)^{2} \tag{15}$$

In batch processing, there are $|B|$ samples exist. The actor policy network updates its strategy using sampled gradients

as follows:

$$\nabla_{\vartheta^{\mu}} D \approx \frac{1}{|B|} \sum \nabla_a Q\left(\mathrm{C}, a|\vartheta^{Q}\right)|_{a=\mu(\mathrm{C})} \nabla_{\vartheta^{\mu}} \mu\left(\mathrm{C}|\vartheta^{\mu}\right) \quad (16)$$

Moreover, stability is particularly crucial in tensor decomposition tasks, as decisions made at each step affect subsequent steps, and any unstable updates may lead to the failure of the entire decomposition process. Therefore, to reduce fluctuations during the training process, make the learning process more stable, and ensure smooth changes in the parameters of the policy network, this study adopts a soft update method, where the target network synchronizes with the current network using the following rule after a specific number of steps:

$$\begin{aligned} \vartheta^{Q'} &\leftarrow \tau\vartheta^{Q} + (1-\tau)\,\vartheta^{Q'} \\ \vartheta^{\mu'} &\leftarrow \tau\vartheta^{\mu} + (1-\tau)\,\vartheta^{\mu'} \end{aligned} \quad (17)$$

Here, τ is a hyperparameter representing the degree of hardness or softness of net-work synchronization. When $\tau = 1$, it is a hard update, and when $0 < \tau < 1$, it is a soft update, we hereby set it to 0.01.

Through the above game process, we can obtain the number of steps required to decompose the convolutional weights at each layer of the CNN model, enabling us to obtain the desired CP rank. Subsequently, we apply the obtained CP rank to the trained CNN model and perform CP decomposition using the CP-alternating least squares (CP-ALS) method [20]. After the decomposition is completed, we obtain a compressed model. Then, a traditional iterative fine-tuning technique is utilized, involving a second round of training by adjusting parameters such as the learning rate and batch size. This enables the network to continuously adapt to the requirements of new tasks, restoring and optimizing its performance. Eventually, the compression of the CNN model is achieved.

IV. EXPERIMENTAL RESULTS

To solve (5), there are multiple methods for estimating the CP rank nowadays [33], [34], only a few are widely adopted, owing to the limitations of various methods. These methods can be classified into two categories. One is to use the "average rank" derived from $\mathcal{N}/3$ or $\mathcal{N}/4$ as the rank for direct CP decomposition, where $\mathcal{N}$ represents the maximum value in the shape of the convolution weights. The other category includes the VBMF algorithm and its improved methods. In the literature [35] and [36], VBMF and its improved methods were used to estimate the CP rank, converting the tensor decomposition problem into a SVD problem for matrices. VBMF weighs the singular values obtained from SVD and solves a quartic equation to determine the importance of each singular value, helping to separate signals from noise and thus determine the rank for CP decomposition. In this study, we employed three aforementioned methods, namely $\mathcal{N}/3$, $\mathcal{N}/4$, and the VBMF algorithm, and the ranks determined by each of these methods were subsequently applied in the CP-ALS method to perform CP decomposition. The DecG framework underwent iterative training for 300,000 epochs using the AMD EPYC-9754 CPU and the NVIDIA GeForce RTX-4090 GPU, with models being saved every 50,000 epochs. Upon completion of training, the results obtained from the DecG framework (see Fig. 4) were thoroughly compared with those from three other methods. Following the completion of decomposition for each method, fine-tuning was iteratively conducted on each model to comprehensively evaluate the performance of the different CP rank selection methods in the experiments, thereby providing a detailed comparison and explanation of the experimental results.

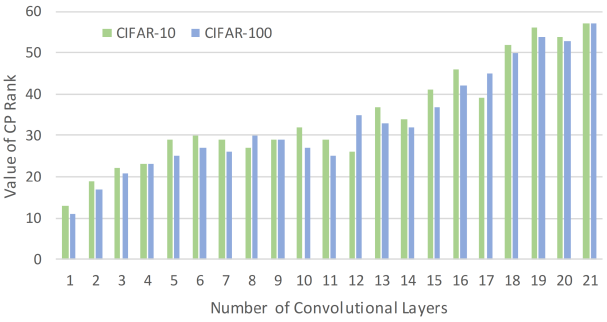


FIGURE 4. The CP rank values of each convolutional layer in the ResNet20 model, which is trained on CIFAR-10 and CIFAR-100 datasets, calculated using the DecG single-player game framework.

To verify the effectiveness of the method proposed in this paper, commonly used models are selected to be validated on the CIFAR-10 and CIFAR-100 datasets. The deep learning framework pytorch was adopted, and NVIDIA GeForce RTX 4090 was used to complete the experimental process. For the comparison experiments, we selected several widely used networks as baselines, including ResNet20, ResNet56, ResNet110 [37], and LeNet.

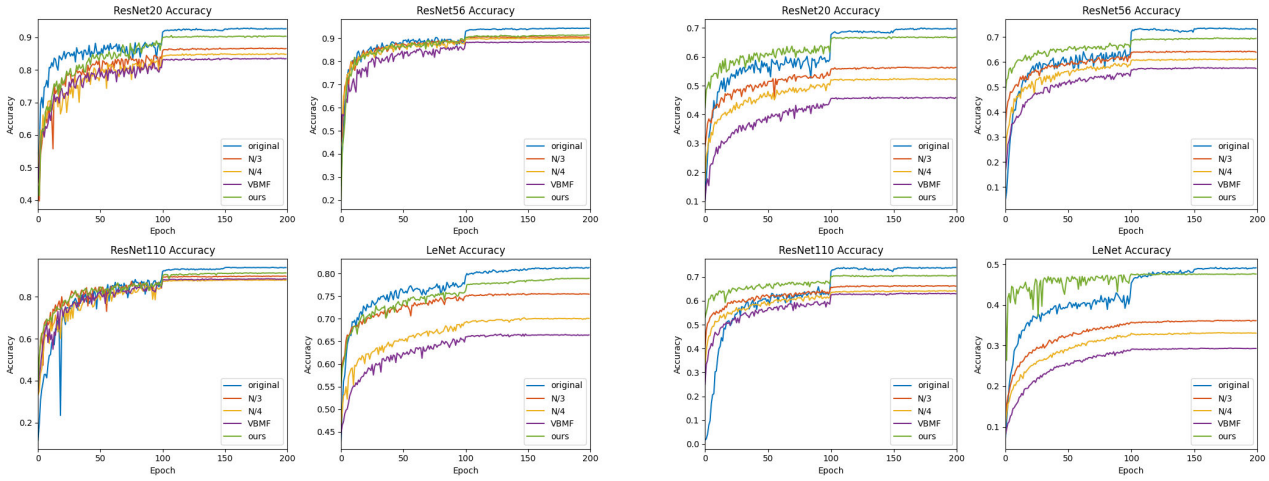
A. EVALUATION CRITERIA

In this study, a novel metric is proposed to comprehensively evaluate the performance of different compression methods. When selecting a compression approach, two critical factors, accuracy and model size, are considered to identify the optimal compression strategy. The Compression Efficiency (CE) is defined as the ratio between the Compression Ratio (CR) and the degree of Accuracy Decline (AD) relative to the original model. Here, the Compression Ratio represents the ratio of the size of the original model to that of the compressed model, calculated using the formula as follows:

$$CR = S_{\text{before}} / S_{\text{after}} \quad (18)$$

where S represents the amount of storage space occupied by the CNN model. A higher CR indicates more thorough compression of the model.

The Accuracy Decline refers to the extent of the accuracy reduction in the compressed model compared with the



(a) CIFAR-10

(b) CIFAR-100

FIGURE 5. The changes in accuracy during the iterative fine-tuning process for the original network models and those decomposed using various rank selection methods on the ResNet20, ResNet56, ResNet110, and LeNet networks. (a)on the CIFAR-10 dataset. (b)on the CIFAR-100 dataset.

original model, calculated using the formula as follows:

$$AD = Accuracy_{\text{before}} - Accuracy_{\text{after}} \tag{19}$$

The calculation formula for the Compression Efficiency can be derived from CR and AD as follows:

$$CE = CR/AD \tag{20}$$

By calculating the Compression Efficiency, the performances of the different compression methods can be directly compared. A higher CE value indicates a more effective model compression whereas maintaining a certain level of accuracy.

Regarding compression, we comprehensively compare different compression methods through Floating Point Operations (Flops), number of parameters, Multiply–Accumulate Operations (MAdd), and final compression ratio. Among them, Flops can be used to measure the computational complexity of a model and are often used indirectly as a standard to assess the speed of neural network models. For convolutional layers, the calculation formula for Flops is as follows:

$$Flops = 2 \cdot \mathrm{H} \cdot \mathrm{W} \cdot \mathrm{C}_{in} \cdot \mathrm{K}^2 \cdot \mathrm{C}_{out} \tag{21}$$

where H and W represent the height and width of the output tensor respectively, C_{in} is the number of channels in the input tensor of the CNN convolutional layer, C_{out} is the number of channels in the output tensor of the CNN convolutional layer, and K is the size of the convolutional kernel. The multiplication by 2 is due to the fact that the number of addition operations is the same as the number of multiplication operations in a convolutional operation. The number of parameters refers to the total number of parameters that need to be learned in the model, which can be used to measure the complexity of a model and reflect the amount of memory required by the model. MAdd involves one multiplication operation and one addition operation, and there is usually a two-fold relationship between MAdd and Flops.

B. ACCURACY COMPARISON

The experiment selected the CIFAR-10 and CIFAR-100 datasets and employed three ResNet networks with varying numbers of convolutional layers along with a LenNet network that only has two convolutional layers. In addition, CP decomposition was implemented using the $\mathcal{N}/3$, $\mathcal{N}/4$, and VBMF algorithms. During the experiment, a learning rate adjustment strategy was adopted. The ResNet networks utilized a learning rate of 0.1 for the first 100 iterations, whereas the LeNet network employed a learning rate of 0.001 in the early stages of training. Subsequently, the learning rate was reduced to 10% of its previous value every 50 iterations (see Fig. 5, Table 1, and Table 2).

In Fig. 5(a), through the analysis of the ResNet20 and LeNet cases on CIFAR-10, we observe that during the first 100 iterations, the convergence speed of the models undergoing CP decomposition during the fine-tuning phase is significantly lower than that of the original models. However, when we apply the same method to deeper network architectures such as ResNet56 and ResNet110, the situation is entirely different, as shown in the figure. In these more complex models, CP decomposition significantly accelerates convergence speed, demonstrating its advantage in handling large-scale networks. By applying the DecG method to determine the rank values, during the process of introducing the CP decomposition, we find that the decomposed model exhibited significantly faster convergence speed compared to other methods. As evident from Fig. 5(b), the same trend is observed on the CIFAR-100 dataset. The ResNet network exhibits a significantly faster convergence speed as the number of layers increases, whereas the LeNet network, despite considerable fluctuations in its model performance, still demonstrates a relatively rapid convergence speed.

TABLE 1. A comparison of the final converged accuracy rates between the original network models and the decomposed models after iterative fine-tuning on the CIFAR-10 dataset.

Model	Method	Accuracy	Model	Method	Accuracy
ResNet20	original	92.9%	ResNet56	original	94.4%
	$\mathcal{N}/3$	86.5%		$\mathcal{N}/3$	90.5%
	$\mathcal{N}/4$	84.8%		$\mathcal{N}/4$	90.0%
	VBMF	83.5%		VBMF	88.4%
	ours	90.3%		ours	91.5%
ResNet110	original	94.0%	LeNet	original	81.3%
	$\mathcal{N}/3$	89.7%		$\mathcal{N}/3$	75.5%
	$\mathcal{N}/4$	87.9%		$\mathcal{N}/4$	70.1%
	VBMF	88.5%		VBMF	66.4%
	ours	91.4%		ours	78.9%

TABLE 2. A comparison of the final converged accuracy rates between the original network models and the decomposed models after iterative fine-tuning on the CIFAR-100 dataset.

Model	Method	Accuracy	Model	Method	Accuracy
ResNet20	original	69.7%	ResNet56	original	73.2%
	$\mathcal{N}/3$	56.3%		$\mathcal{N}/3$	64.2%
	$\mathcal{N}/4$	52.1%		$\mathcal{N}/4$	61.2%
	VBMF	45.9%		VBMF	57.6%
	ours	66.8%		ours	69.5%
ResNet110	original	74.0%	LeNet	original	49.2%
	$\mathcal{N}/3$	66.2%		$\mathcal{N}/3$	36.1%
	$\mathcal{N}/4$	64.1%		$\mathcal{N}/4$	33.0%
	VBMF	63.0%		VBMF	29.3%
	ours	70.5%		ours	47.6%

Moreover, the rank values obtained through the DecG method enable the decomposed model to maintain a significant advantage compared to other methods. In most cases, the initial accuracy of the four network models after decomposition is higher than before decomposition. CP decomposition preserves some of the reasoning capabilities of the original model, improving the initial performance of the model. These experimental results indicate that the application effects of CP decomposition vary across different network architectures and under varying rank settings, and furthermore, it is more suitable for deep CNN models. For CNN models with varying degrees of complexity, detailed adjustments and optimizations must be made for specific tasks and network structures.

In Table 1 and Table 2, after the CP-decomposed models converge through iterative fine-tuning, we compare the accuracy of each model. The experimental results reveal that, when assessing the accuracy of ResNet20 on the CIFAR-10 and CIFAR-100 classification datasets, although our CP rank selection method achieves a decrease in accuracy of 2.6% and 2.9% respectively compared to the original model, it still outperforms other methods. Specifically, when compared to the $\mathcal{N}/3$, $\mathcal{N}/4$, and VBMF methods, our accuracy is improved by 3.8%, 5.5%, and 6.8% on the CIFAR-10 dataset, and by 10.5%, 14.7%, and 20.9% on the CIFAR-100 dataset.For LeNet, our algorithm only exhibits a decrease in accuracy of 2.4% and 1.6% compared to the original model. In deeper network architectures such as ResNet56 and ResNet110, our method continues to demonstrate superior performance, not only maintaining a leading position in effectiveness but also exhibiting excellent robustness, with accuracy rates surpassing the other three methods. For instance, on the CIFAR-10 dataset using ResNet110, our method outperforms the other methods by 1.7%, 3.5%, and 2.9%, while for CIFAR-100, the improvements are 4.3%, 6.4%, and 7.5%. Simultaneously, as the network architecture gradually deepens, the accuracy decline of the VBMF method gradually diminishes. It can be observed that, when performing recognition on the CIFAR-10 dataset using the ResNet110 network architecture, the VBMF method even surpasses the $\mathcal{N}/4$ method.

In summary, although CP decomposition may lead to a slight decrease in accuracy in some scenarios, the method proposed in this study demonstrates superior performance in terms of maintaining accuracy compared with other methods. Notably, whereas VBMF may be more suitable for CP rank selection in more complex CNN models, our method generally improves the accuracy and exhibits superior robustness. These results provide valuable guidance for further optimizing the application of CP decomposition in convolutional neural networks.

C. MODEL COMPRESSION EXPERIMENT

This study evaluates the compression performance of different methods on multiple network architectures. As shown in Table 3 and Table 4, on ResNet20, although the

TABLE 3. Experimental comparison results of model compression on the CIFAR-10 dataset.

Model	Method	Flops	parameter	MAdd	CR	CE
ResNet20	original	43.62M	1.12M	86.73M	1×	—
	$\mathcal{N}/3$	4.93M	0.10M	9.11M	5.99×	0.936
	$\mathcal{N}/4$	4.03M	0.08M	7.36M	6.66×	0.822
	VBMF	3.74M	0.07M	6.79M	7.09×	0.754
	ours	14.74M	0.25M	28.12M	3.34×	**1.283**
ResNet56	original	129.76M	3.35M	258.15M	1×	—
	$\mathcal{N}/3$	12.78M	0.29M	23.62M	6.54×	1.677
	$\mathcal{N}/4$	10.47M	0.23M	19.12M	7.31×	1.661
	VBMF	10.95M	0.22M	20.04M	7.45×	1.242
	ours	18.54M	0.40M	34.83M	5.42×	**1.871**
ResNet110	original	258.97M	6.70M	515.27M	1×	—
	$\mathcal{N}/3$	24.55M	0.57M	45.4M	6.73×	1.565
	$\mathcal{N}/4$	20.12M	0.45M	36.76M	7.51×	1.231
	VBMF	31.32M	0.79M	58.61M	5.55×	1.009
	ours	49.09M	0.91M	92.98M	4.92×	**1.893**
LeNet	original	2.36M	0.46M	4.67M	1×	—
	$\mathcal{N}/3$	0.33M	0.41M	0.58M	1.11×	0.191
	$\mathcal{N}/4$	0.29M	0.41M	0.51M	1.11×	0.099
	VBMF	0.26M	0.41M	0.47M	1.11×	0.074
	ours	0.83M	0.42M	1.58M	1.03×	**0.429**

TABLE 4. Experimental comparison results of model compression on the CIFAR-100 dataset.

Model	Method	Flops	parameter	MAdd	CR	CE
ResNet20	original	43.62M	1.14M	86.74M	1×	—
	$\mathcal{N}/3$	4.93M	0.13M	9.12M	5.51×	0.411
	$\mathcal{N}/4$	4.03M	0.10M	7.37M	6.05×	0.344
	VBMF	3.60M	0.08M	6.51M	6.98×	0.293
	ours	14.07M	0.27M	26.82M	3.30×	**1.138**
ResNet56	original	129.76M	3.37M	258.16M	1×	—
	$\mathcal{N}/3$	12.78M	0.31M	23.63M	6.34×	0.704
	$\mathcal{N}/4$	10.47M	0.25M	19.13M	7.06×	0.588
	VBMF	10.46M	0.19M	19.04M	7.99×	0.512
	ours	23.01M	0.49M	43.50M	4.79×	**1.295**
ResNet110	original	258.97M	6.72M	515.29M	1×	—
	$\mathcal{N}/3$	24.56M	0.59M	45.41M	6.62×	0.849
	$\mathcal{N}/4$	20.13M	0.47M	36.77M	7.37×	0.744
	VBMF	27.37M	0.45M	50.68M	7.53×	0.685
	ours	50.82M	0.98M	96.34M	4.81×	**1.374**
LeNet	original	2.37M	0.49M	4.68M	1×	—
	$\mathcal{N}/3$	0.33M	0.44M	0.60M	1.10×	0.084
	$\mathcal{N}/4$	0.30M	0.44M	0.53M	1.11×	0.068
	VBMF	0.27M	0.44M	0.48M	1.11×	0.055
	ours	0.85M	0.45M	1.61M	1.09×	**0.681**

VBMF algorithm achieves significant compression with a compression ratio of up to 7.09 on the CIFAR-10 dataset, according to the analysis in Section III-B, the significant decrease in accuracy leads to a relatively low compression benefit of only 0.754. However, given the complexity of the CIFAR-100 dataset, the decomposition of the model would lead to a more significant drop in accuracy. In the case of the Resnet20 model, when the compression ratio is 6.98, the compression efficiency is only 0.293. In contrast, although our proposed method has a slightly smaller compression ratio, it achieves higher compression efficiency while maintaining a high accuracy, reaching 1.283 and 1.138 for the two datasets respectively.

Furthermore, in network architectures such as ResNet56, ResNet110, and LeNet, we employed the DecG method for rank selection and found that this method performed better in terms of compression efficiency for these networks, with minimal influence from the dataset. This validates the versatility and efficacy of our proposed approach, which is capable of achieving efficient compression whereas preserving

a stable model performance across diverse network architectures.

After comprehensively comparing the compression results of various methods across different network architectures, we observed an increasingly significant upward trend in both the compression ratio and compression benefit as the number of layers in the network model increased (i.e., as the model complexity increased). Specifically, in simpler networks such as LeNet, the compression effect of CP decomposition is not ideal, with an average compression benefit of merely 0.176 and 0.222 across two datasets, after a comprehensive evaluation of four compression strategies. In contrast, the average compression benefit of ResNet20 increased to 0.949 and 0.547, demonstrating a certain potential despite room for further optimization. During a thorough analysis of more intricate network architectures such as ResNet56 and ResNet110, we observed that the compression benefit of CP decomposition was significantly enhanced. Notably, in the case of ResNet56, as the number of parameters and model layers increased, the average compression benefit of the model after CP decomposition reached a remarkable value of 1.613 and 0.775, demonstrating exceptional performance. However, it is worth noting that despite the deeper network layers of ResNet110, its average compression efficiency on the CIFAR-10 dataset slightly decreased to 1.424. Furthermore, it still outperformed ResNet56 on the CIFAR-100 dataset, achieving a result of 0.913. This difference may stem from the insufficient robustness of the other three rank selection methods when faced with deeper network architectures.

Against this backdrop, the rank selection method based on reinforcement learning proposed in this study demonstrated a remarkable performance. Compared with other methods, this approach not only achieves the highest compression benefit but also exhibits a higher growth trend in compression benefit as the number of network layers increases. This demonstrated the excellent robustness and wide applicability of the proposed method. Additionally, as clearly shown in the table, after CP decomposition, the number of parameters, floating-point operations, and multiply-add operations in various networks decreased, further validating the potential of CP decomposition to accelerate model inference.

V. CONCLUSION

In the work presented in this paper, a novel method for CP rank selection based on deep reinforcement learning is proposed. This method fully leverages the advantages of the DDPG algorithm and achieves automation and intelligence in model rank selection through the design of the DecG single-player game framework. In the process of continuous exploitation and exploration, this study introduces a pre-trained model and designs a deterministic policy based on singular value decomposition, significantly improving the efficiency of the model in finding ideal rank selection results. This method can accurately target any convolutional layer and automatically determine the appropriate rank based on the characteristics of its weight tensor, thereby optimizing the compression benefit and performance of the model. To validate the effectiveness of the proposed method, multiple CNN models were selected from the CIFAR-10 dataset and were subjected to extensive testing. Compared with current mainstream methods, this method demonstrated notable advantages across various performance metrics. This offers a novel approach to the compression and acceleration of the convolutional neural networks.

Notably, the CP-decomposed models have undergone initial experimental validation on PC terminals, achieving significant reductions in storage requirements and multiplication-addition operations for various CNN models. However, they have not yet been deployed and tested on actual edge computing devices. Therefore, the next research direction will primarily focus on integrating this CP decomposition method with edge computing devices, such as Field Programmable Gate Arrays (FPGAs), to achieve efficient resource utilization. We aim to harness the parallel processing capabilities and the flexibility of configurable multipliers inherent in FPGAs by developing a CNN model accelerator grounded in CP decomposition. This accelerator will parallelize the decomposed convolutional computations, possibly implementing a pipelined architecture that resolves conflicts and dependencies among tasks, and optimizing data access patterns, to facilitate the efficient deployment and utilization of CP-decomposed CNN models on edge devices. A current challenge lies in the altered kernel dimensions post-decomposition, necessitating careful consideration in the design of convolutional computing units to ensure efficient computations.

REFERENCES

[1] W. Zhou, H. Wang, and Z. Wan, "Ore image classification based on improved CNN," *Comput. Electr. Eng.*, vol. 99, Apr. 2022, Art. no. 107819.

[2] C. Farabet, C. Couprie, L. Najman, and Y. LeCun, "Learning hierarchical features for scene labeling," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 35, no. 8, pp. 1915–1929, Aug. 2013.

[3] N. O'Mahony, S. Campbell, A. Carvalho, S. Harapanahalli, G. Velasco-Hernandez, L. Krpalkova, D. Riordan, and J. Walsh, "Deep learning vs. traditional computer vision," in *Proc. Adv. Comput. Vis. Conf. (CVC)*, vol. 11, May 2020, pp. 128–144.

[4] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proc. IEEE*, vol. 86, no. 11, pp. 2278–2324, Nov. 1998.

[5] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," 2014, *arXiv:1409.1556*.

[6] Q. Zhang, M. Zhang, T. Chen, Z. Sun, Y. Ma, and B. Yu, "Recent advances in convolutional neural network acceleration," *Neurocomputing*, vol. 323, pp. 37–51, Jan. 2019.

[7] G. Rong, Y. Xu, X. Tong, and H. Fan, "An edge-cloud collaborative computing platform for building AIoT applications efficiently," *J. Cloud Comput.*, vol. 10, no. 1, p. 36, Dec. 2021.

[8] H. Gao, Y. Tian, and F. Xu, "A review of deep learning model compression and acceleration," *Softw. J.*, vol. 32, no. 1, pp. 68–92, 2021.

[9] X. Long, "Research on compression and acceleration techniques for deep convolutional network models," Ph.D. dissertation, Dept. Control Sci. Eng., Nat. Univ. Defense Technol., Changsha, China, 2020.

[10] D. Song, P. Zhang, and F. Li, "Speeding up deep convolutional neural networks based on tucker-CP decomposition," in *Proc. 5th Int. Conf. Mach. Learn. Technol.*, Jun. 2020, pp. 1–11.

[11] J. Håstad, "Tensor rank is NP-complete," in *Proc. 16th Int. Colloq. Automata, Lang. Programming*, Stresa, Italy, 1989, pp. 451–460.
[12] M. Denil, B. Shakibi, L. Dinh, M. Ranzato, and N. de Freitas, "Predicting parameters in deep learning," in *Proc. 26th Int. Conf. Neural Inf. Process. Syst.*, 2013, pp. 2148–2156.
[13] E. L. Denton, W. Zaremba, J. Bruna, Y. LeCun, and R. Fergus, "Exploiting linear structure within convolutional networks for efficient evaluation," in *Proc. Int. Conf. Adv. Neural Inf. Process. Syst. (NIPS)*, 2014, pp. 1269–1277.
[14] B. Praggastis, D. Brown, C. O. Marrero, E. Purvine, M. Shapiro, and B. Wang, "The SVD of convolutional weights: A CNN interpretability framework," 2022, *arXiv:2208.06894*.
[15] Y. Li, S. Lin, J. Liu, Q. Ye, M. Wang, F. Chao, F. Yang, J. Ma, Q. Tian, and R. Ji, "Towards compact CNNs via collaborative compression," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2021, pp. 6434–6443.
[16] S. Swaminathan, D. Garg, R. Kannan, and F. Andres, "Sparse low rank factorization for deep neural network compression," *Neurocomputing*, vol. 398, pp. 185–196, Jul. 2020.
[17] Y. D. Kim, E. Park, and S. Yoo, "Compression of deep convolutional neural networks for fast and low power mobile applications," in *Proc. Int. Conf. Comput. Vis.*, 2015, pp. 576–584.
[18] T. E. Rasmussen, L. K. H. Clemmensen, and A. Baum, "Compressing CNN kernels for videos using tucker decompositions: Towards light-weight CNN applications," in *Proc. Northern Lights Deep Learn. Workshop*, vol. 3, 2022. [Online]. Available: https://orbit.dtu.dk/en/publications/compressing-cnn-kernels-for-videos-using-tucker-decompositions-to and https://arxiv.org/abs/2203.07033
[19] M. Yin, S. Liao, X.-Y. Liu, X. Wang, and B. Yuan, "Compressing recurrent neural networks using hierarchical tucker tensor decomposition," 2020, *arXiv:2005.04366*.
[20] T. G. Kolda and B. W. Bader, "Tensor decompositions and applications," *SIAM Rev.*, vol. 51, no. 3, pp. 455–500, Aug. 2009.
[21] M. Astrid and S.-I. Lee, "CP-decomposition with tensor power method for convolutional neural networks compression," in *Proc. IEEE Int. Conf. Big Data Smart Comput. (BigComp)*, Feb. 2017, pp. 115–118.
[22] Y. Ji and Q. Wang, "Fast CP-compression layer: Tensor CP-decomposition to compress layers in deep learning," *IET Image Process.*, vol. 16, no. 9, pp. 2535–2543, Jul. 2022.
[23] Q. Zhao, L. Zhang, and A. Cichocki, "Bayesian CP factorization of incomplete tensors with automatic rank determination," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 37, no. 9, pp. 1751–1763, Sep. 2015.
[24] Q. Zhao, G. Zhou, L. Zhang, A. Cichocki, and S.-I. Amari, "Bayesian robust tensor factorization for incomplete multiway data," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 27, no. 4, pp. 736–748, Apr. 2016.
[25] S. Nakajima, R. Tomioka, and M. Sugiyama, "Perfect dimensionality recovery by variational Bayesian PCA," in *Proc. 25th Int. Conf. Neural Inf. Process. Syst.*, vol. 1, 2012, pp. 971–979.
[26] M. Zhou, Y. Liu, Z. Long, L. Chen, and C. Zhu, "Tensor rank learning in CP decomposition via convolutional neural network," *Signal Process., Image Commun.*, vol. 73, pp. 12–21, Apr. 2019.
[27] A. Fawzi, M. Balog, A. Huang, T. Hubert, B. Romera-Paredes, M. Barekatain, A. Novikov, F. J. R. Ruiz, J. Schrittwieser, G. Swirszcz, D. Silver, D. Hassabis, and P. Kohli, "Discovering faster matrix multiplication algorithms with reinforcement learning," *Nature*, vol. 610, no. 7930, pp. 47–53, Oct. 2022. [Online]. Available: https://www.nature.com/articles/s41586-022-05172-4
[28] L. Ardon, "Reinforcement learning to solve NP-hard problems: An application to the CVRP," 2022, *arXiv:2201.05393*.
[29] Y. Hou, L. Liu, Q. Wei, X. Xu, and C. Chen, "A novel DDPG method with prioritized experience replay," in *Proc. IEEE Int. Conf. Syst., Man, Cybern. (SMC)*, Oct. 2017, pp. 316–321.
[30] S. Liu, G. Cao, Y. Liu, Y. Li, C. Wu, and X. Xi, "Mix-up consistent cross representations for data-efficient reinforcement learning," in *Proc. Int. Joint Conf. Neural Netw. (IJCNN)*, Jul. 2022, pp. 12686–12699.
[31] T. Rajapakshe, S. Latif, R. Rana, S. Khalifa, and B. W. Schuller, "Deep reinforcement learning with pre-training for time-efficient training of automatic speech recognition," 2020, *arXiv:2005.11172*.
[32] J.-J. Kim, S.-H. Cha, M. Ryu, and M. Jo, "Pre-training framework for improving learning speed of reinforcement learning based autonomous vehicles," in *Proc. Int. Conf. Electron., Inf., Commun. (ICEIC)*, Jan. 2019, pp. 1–2.
[33] M. Ashraphijuo, X. Wang, and V. Aggarwal, "An approximation of the CP-rank of a partially sampled tensor," in *Proc. 55th Annu. Allerton Conf. Commun., Control, Comput. (Allerton)*, Oct. 2017, pp. 604–611.
[34] Q. Shi, H. Lu, and Y.-M. Cheung, "Tensor rank estimation and completion via CP-based nuclear norm," in *Proc. ACM Conf. Inf. Knowl. Manage.*, Nov. 2017, pp. 949–958.
[35] M. Astrid, S.-I. Lee, and B.-S. Seo, "Rank selection of CP-decomposed convolutional layers with variational Bayesian matrix factorization," in *Proc. 20th Int. Conf. Adv. Commun. Technol. (ICACT)*, Feb. 2018, pp. 347–350.
[36] T. Kim, J. Lee, and Y. Choe, "Bayesian optimization-based global optimal rank selection for compression of convolutional neural networks," *IEEE Access*, vol. 8, pp. 17605–17618, 2020.
[37] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 770–778.

SHAOSHUANG ZHANG is currently pursuing the degree with Shandong University of Technology. She is engaged in the research of artificial intelligence. Her specific research direction is to implement compression and acceleration of neural network models.

ZHAO LI received the Ph.D. degree from Beihang University. He is currently an Associate Professor with Shandong University of Technology. He is mainly engaged in artificial intelligence and edge computing. He has completed a Ph.D. research start-up fund project of Shandong University of Science and Technology and participated in two Shandong Provincial Natural Science Foundation projects, including one Shandong Provincial Natural Science Foundation general fund project. In recent years, he has published eight articles as the first author and obtained two authorized invention patents.

WENLONG LIU is currently pursuing the degree with Shandong University of Technology. He is engaged in the research of artificial intelligence in computer vision. His specific research direction is to object detection.

JIAQI ZHAO is currently pursuing the degree with Shandong University of Technology. She is engaged in the research of artificial intelligence. Her specific research direction is to implement compression and acceleration of neural network models.

TING QIN is currently pursuing the degree with Shandong University of Technology. She is engaged in the research of artificial intelligence. Her specific research direction is to implement compression and acceleration of neural network models.