ISSN 2616-5775 Vol. 8, Issue 7: 79-86, DOI: 10.25236/AJCIS.2025.080710

Structured Pruning Based on Reinforcement Learning for CNN

Qianxi Li[1,a,*], Wenhui Zhang[1,b]

[1]School of Information Science and Engineering, Chongqing Jiaotong University, Chongqing, 400074, China
[a]lqxdyx0121@163.com, [b]17385339798@163.com
**Corresponding author*

Abstract: *In recent years, deep learning models have demonstrated excellent performance in complex tasks, but their large number of parameters and high computational costs have limited their application in resource-constrained scenarios. This paper proposes a structured pruning method based on reinforcement learning (TD3 algorithm), which performs structured pruning on a group-by-group basis to balance model compression efficiency and performance retention. The TD3 agent takes the parameter states of each group as observation inputs, dynamically adjusts the pruning rate of each group as actions, and designs a multi-objective reward function based on model accuracy,FLOPs, and the number of parameters to achieve autonomous optimization of pruning strategies. Experiments on ResNet56 and VGG19 with the CIFAR-100 dataset show that this method maintains high classification accuracy while significantly reducing parameters and computational complexity. Compared with traditional pruning methods, it is more adaptive and provides an effective solution for model deployment in resource-constrained environments.*

Keywords: *Structured Pruning, Reinforcement Learning, TD3, Model Compression*

1. Introduction

In recent years, Deep Learning has made remarkable progress in various fields and applications such as Computer Vision (CV)[1], Natural Language Processing (NLP)[2], and Audio Signal Processing (ASP) [3]. Although Deep Neural Networks (DNNs) have achieved remarkable success in various fields, their performance heavily relies on extensive model parameters and high computational costs. For instance, the widely used ResNet-50 occupies more than 95 MB of memory and contains more than 23 million parameters [4]. Although the huge number of parameters significantly improves the model's task processing performance, it also brings high computational costs and long inference times, seriously limiting the deployment and application of deep learning models in resource-constrained scenarios [5]. To address this challenge, model compression methods have emerged, aiming to reduce model parameters and computational complexity while minimizing performance degradation, thereby improving operational efficiency .

At present, the mainstream methods of model compression mainly include pruning[6], quantization[7], knowledge distillation[8], and low-rank decomposition[9], etc. Quantization converts high-precision floating-point parameters into low-precision integer or binary-valued parameters, potentially sacrificing model accuracy.Low-rank decomposition method decomposes a high-dimensional weight matrix into the product of multiple low-rank matrices, however, the effect of low-rank decomposition depends on the accuracy and method of matrix decomposition. In contrast, the model pruning directly reduces the complexity of the model by removing unimportant connections, neurons, or parameters in the neural network. While achieving a significant compression effect, it can better maintain the model performance, thus attracting extensive attention from researchers.

Model pruning methods are mainly categorized into unstructured pruning and structured pruning. Unstructured pruning will disrupt the structural rules of the model. It requires special hardware acceleration and is difficult to deploy. In contrast, structured pruning aims to preserve the regularity of the network architecture by eliminating entire neurons, convolutional kernels, or channels, thereby enabling broader practical applications. Fang et al.[10] decomposed the substructures in the network into multiple isomorphic groups, and independently performed importance evaluation and pruning within each group to achieve group-level pruning.Liu et al.[11] used the reparameterization to merge adjacent

ISSN 2616-5775 Vol. 8, Issue 7: 79-86, DOI: 10.25236/AJCIS.2025.080710

layers, compressed the multi-branch structure into a single-path convolutional layer, and utilized structure reparameterization to solve the compatibility problem of the normalization layer. Notably, traditional structured pruning methods predominantly rely on manual experience when determining pruning strategies and it is difficult to adapt to different models and tasks.

To overcome these limitations, this paper proposes a reinforcement-learning-based model pruning method that leverages Twin Delayed Deep Deterministic Policy Gradient (TD3). First, the model is divided into different groups according to the dependency relationships among its layers[12]. Subsequently, the TD3 agent is applied to the pruning process, taking the set of parameter states of each group as the observation input for the agent, and defining the group - level pruning operation as the agent's action. A reward function is constructed based on the performance change of the pruned model, enabling the agent to autonomously explore the optimal group - based pruning strategy through interaction and learning with the model environment. Compared with traditional methods, this scheme can adaptively perform group - based pruning on the model, dynamically adjust the pruning intensity of each group, achieve a higher compression ratio, and minimize accuracy loss.

2. Related Work

2.1 Structured Pruning

Structured pruning is straightforward in operation, capable of preserving the original network structure, which facilitates subsequent analysis and optimizationand is easy to implement and deploy. Consequently, researchers have conducted extensive investigations into structured pruning.

Wang et al.[13] statistically modeled the network pruning problem from the perspective of reducing redundancy, developing a pruning method that identifies the structural redundancy of CNNs and prunes the filters in selected layers with the highest redundancy. Benbaki et al.[14] considered the comprehensive effects of pruning multiple weights subject to sparsity constraints, performing combinatorial optimization updates on a memory - friendly representation of the local quadratic approximation of the loss function. Existing methods lack quantifiable metrics to estimate the compressibility of sub - networks during each pruning iteration. To address this, Diao et al.[15] introduced the PQ index to measure the potential compressibility of deep neural networks and utilized it to develop a sparsity - aware adaptive pruning algorithm.

Currently, the absence of an automated mechanism in existing structured pruning approaches requires researchers to invest substantial time in manual parameter tuning and experimental verification. Therefore, designing a method that can automatically and efficiently determine the optimal pruning rate is essential to propel structured pruning technology towards intelligence and automation.

2.2 Deep Reinforcement Learning

Deep reinforcement learning (DRL) integrates the powerful feature extraction capabilities of deep learning with the decision-making optimization capabilities of reinforcement learning, demonstrating significant advantages in handling high-dimensional and complex state spaces and action spaces. Deep learning automatically extracts abstract feature representations from raw data through multi-layer neural networks, providing rich information inputs for reinforcement learning. Reinforcement learning, meanwhile, optimizes an agent’s policy via interaction with the environment and reward signals, enabling the agent to make optimal decisions in complex environment.

Deep Q-Network (DQN) [16] is one of the classic algorithms of deep reinforcement learning.It combines a convolutional neural network with Q-learning to handle reinforcement learning tasks with complex inputs such as images, achieving end-to-end deep reinforcement learning for the first time and laying the foundation for the development of subsequent algorithms. Asynchronous Advantage Actor-Critic (A3C)[17] accelerates training efficiency and reduces variance by asynchronously executing multiple agents’ learning processes, performing excellently in various Atari game tasks. Deep Deterministic Policy Gradient (DDPG)[18] tackles continuous action space problems using an Actor-Critic architecture, coupled with target networks and experience replay mechanisms, significantly enhancing algorithm stability and convergence speed. TD3[19] improves upon DDPG by introducing a dual Q-network structure to mitigate Q-value overestimation, target policy smoothing regularization to reduce policy update fluctuations, and delayed update mechanisms to decrease network parameter update frequency. These enhancements further boost the algorithm’s stability and performance in continuous

ISSN 2616-5775 Vol. 8, Issue 7: 79-86, DOI: 10.25236/AJCIS.2025.080710

control tasks.

Traditional pruning methods typically rely on fixed rules or manually set thresholds, struggling to adapt to diverse model architectures and task requirements. In contrast, DRL agents dynamically learn optimal pruning strategies through interaction with the model environment, adjusting pruning intensity and methods based on the model's real-time state. This feature enables DRL-based pruning methods to more effectively balance model compression and performance retention across complex and varied models and tasks. Therefore, this paper employs a TD3-based reinforcement learning approach to pruning, minimizing manual intervention and enhancing automation.

3. Method

Figure 1 provides an overview of the automated model pruning framework based on the TD3 agent. This framework formulates the model pruning process as a reinforcement learning problem. The TD3 agent dynamically searches for the optimal pruning rate on a group-by-group basis, ensuring model accuracy while achieving efficient compression of computational resources. It overcomes the limitations of traditional methods and improves the efficiency of model deployment.

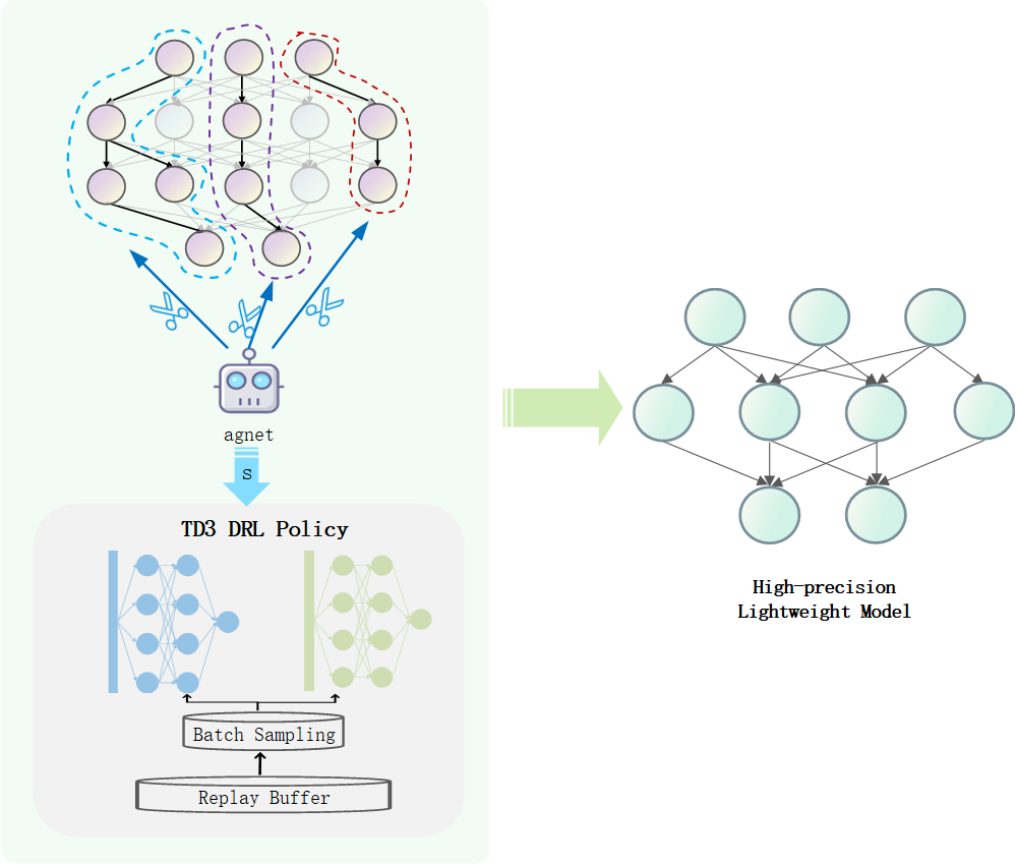

Figure 1: Reinforcement Learning Pruning Framework.

3.1 Grouping Method

This paper introduces the automatic model grouping method proposed by Wang et al. [12]. This method uses dependency graphs to model the dependencies of each layer in the convolutional network. The construction and grouping process of this method does not require human intervention and can avoid disrupting the complex coupling relationships between models during pruning.

3.1.1 Definition

First, a Convolutional Neural Network (CNN) is divided into prunable layers and non-prunable layers.Prunable layers include convolutional layers and linear layers, denoted as $\{\mathcal{L}_i^p, 1 \le i \le L_c + L_f\}$, where L_c and L_f represent the number of convolutional layers and linear layers, respectively. Non-

ISSN 2616-5775 Vol. 8, Issue 7: 79-86, DOI: 10.25236/AJCIS.2025.080710

prunable layers encompass activation layers, pooling layers, and batch normalization layers. The pruning operation of a batch normalization layer is triggered only when its preceding convolutional layer or linear layer is pruned.

3.1.2 Methods for Constructing Dependency Graph

The construction process of the dependency graph can be categorized into two types: basic dependency detection and special connection detection. The detailed implementations are as follows:

Basic Dependency Detection: Input an example image, and extract the input tensors $\{T_l^{in}\}$ and output tensors $\{T_l^{out}\}$ of each layer. If the output tensor of layer $\mathcal{L}_i$ is directly used as the input of layer $\mathcal{L}_j$, a basic dependency relationship is established as shown in Equation (1):

$$DG_{(i,j)}(k_{out}) = [k_{out}], \quad \forall 1 \le k_{out} \le N_i^{out} \tag{1}$$

Special Connection Detection: Special connection detection is further divided into the following three types:

Flattening: If the input tensor T_l^{in} fails to match, it may be due to a flattening operation. In this case, detect the output tensor T_a^{out} after flattening, and construct the dependency relationship as shown in Equation (2), where A_{out} represents the output region area:

$$DG_{(i,j)}(k_{out}) = [k_{out} \cdot A_{out}, (k_{out}+1) \cdot A_{out}] \tag{2}$$

Concatenation: If there exists a T_b^{out} such that $T_a^{in} = \text{concat}(T_b^{out}, T_j^{out})$, the dependency relationship is given by Equation (3), where i_b is the starting index of the concatenation:

$$DG_{(b,a)}(k_{out}) = [i_b + k_{out}] \tag{3}$$

Residual Connection: When $T_a^{in} = T_b^{out} + T_c^{out}$, the dependency relationships are expressed as Equation (4).

$$DG_{(b,a)}(k_{out}) = [k_{out}], \quad DG_{(c,a)}(k_{out}) = [k_{out}] \tag{4}$$

Similarly, the dependency relationships of the Squeeze-and-Excitation module (multiplication operation) are the same as those of the residual connection.

3.1.3 Grouping Process

First, input the sample images to obtain the input tensor T_l^{in} and output tensor T_l^{out} of each layer. Then, detect basic dependencies by identifying the inter-layer dependency relationships of direct connections through tensor matching. Next, handle special connections:Flattening operation: Establish the channel mapping relationship based on the comparison of flattened tensors.Concatenation/Residual connections: Construct synchronous pruning constraints according to the tensor operation logic.Finally, group the pruning sets: Divide the layers that require synchronous pruning into non-overlapping sets to ensure that all layers within the same group can complete the pruning operation collaboratively.

3.2 TD3 Reinforcement Learning Pruning Framework

This method uses reinforcement learning to search for the optimal compression rate of the model in the action space. This section will detail the settings of each part of the reinforcement learning.

3.2.1 State Space

For each group of the model observed by the agent, we have 11 features to represent the state s_g, defined as Equation (5):

$$s_g = \left[g, l_g, c_{in}, c_{out}, k_s, h, w, FLOPs_r, Params_r, acc, a_{g-1}\right] \tag{5}$$

Where g and l_g represent the group index and the intra-group layer index respectively, and

ISSN 2616-5775 Vol. 8, Issue 7: 79-86, DOI: 10.25236/AJCIS.2025.080710

c_{in}, c_{out} are the number of input channels and output channels of the l-th layer; k_s, h, w represent the size, length, and width of the convolutional kernel respectively. The dimension of the kernel is $c_{in} \times c_{out} \times k \times k$, and the input is $c_{out} \times h \times w$.

$FLOPs_r = 1 - FLOPs_{current} / FLOPs_{origin}$ is the reduction ratio of the FLOPs of the model after the current round of pruning operation to the FLOPs of the original model. $Params_r = 1 - Params_{current} / Params_{origin}$ origin is the reduction ratio of the Params of the model after the current round of pruning operation to the Params of the original model; acc is the precision of the current model on the validation set; a_{g-1} are scaled within the range of $(0,1)$ before being passed to the agent. These features are indispensable for different convolutional layers in the intelligent area.

3.2.2 Action Space

Action space $a_g \in \mathcal{A}$ represents the pruning rate of the g-th group, satisfying a $a_g \in (0,1)$. When the traditional discrete space is used as a coarse - grained action space for model compression, due to the insufficiently precise control of sparsity, the number of action combinations grows exponentially, making effective exploration extremely difficult. Therefore, this paper adopts the continuous action space $a_g \in (0,1)$, which can realize more refined and accurate model compression operations.

3.2.3 Rewards

A multi - objective fusion reward function R is designed. This reward function comprehensively considers multiple key factors during the pruning process. It aims to guide the intelligent agent to explore a better pruning strategy while optimizing the model performance, and achieve a balance among model accuracy, computational efficiency, and the number of parameters. It is defined as Equation (6):

$$R = \alpha \cdot (\text{acc}_{current} - \text{acc}_{pre}) - \beta \cdot \frac{FLOPs_{pre} - FLOPs_{current}}{FLOPs_{pre}} - \gamma \cdot \frac{Params_{pre} - Params_{current}}{Params_{pre}} \tag{6}$$

The first term is the difference in model accuracy between the current pruned model and the model from the previous round. α is the accuracy weight coefficient, ensuring that the intelligent agent emphasizes accuracy maintenance during pruning. Usually, the model accuracy shows a downward trend after the pruning operation. However, some studies have shown that the accuracy may increase after model compression. Therefore, for the intelligent agent, a reward is given for accuracy improvement, and a penalty is imposed for accuracy decline. The second and third terms are the reduction ratios of the computational load and the number of parameters, respectively, aiming for the maximum compression ratio. In this study, FLOPs and Params are used as rewards, so a ratio - based quantification method is adopted to return values to the agent. β and γ are the corresponding weight coefficients, and different target priorities can be set according to application scenarios. For example, if the mobile terminal focuses on computational load optimization, β can be increased. The weight coefficients satisfy $\alpha + \beta + \gamma = 1$, and they achieve a balance between the improvement of accuracy and the reduction of computational cost. Finally, the accuracy of the compressed model is evaluated on the validation set, and rewards are provided to the agent according to the reward Equation (6).

3.3 Pruning Process of TD3 Agent

3.3.1 Action Generation and Pruning Operation

Given the current normalized state s, the actor network generates a group-level pruning action a_g, where a_g represents the uniform pruning rate for the g-th group, and $0 < a_g < 1$. The specific execution steps are as follows:

Locate Layer Sets by Group: Based on the grouping results of the dependency graph, determine the set of all network layers $L_g = \{L_{g,1}, L_{g,2}, \ldots, L_{g,n}\}$ (where n is the number of layers in the group) included in the g-th group.

Uniform Pruning within the Group: For each layer $L_{g,i}$ in the group, apply the same pruning rate

ISSN 2616-5775 Vol. 8, Issue 7: 79-86, DOI: 10.25236/AJCIS.2025.080710

a_g. Taking a convolutional layer as an example, assume its weight tensor is $W \in \mathbb{R}^{C_{out} \times C_{in} \times K \times K}$ (C_{out} is the number of output channels, C_{in} is the number of input channels, and K is the size of the convolutional kernel). The number of channels retained after pruning is calculated as Equation (7):

$$C' = C_{out} \times (1 - a_g) \tag{7}$$

The new weight tensor is obtained by retaining the first C' channels.

Channel Importance Ranking: Calculate the importance score of each channel using the following formula and sort them in descending order according to the $L1$-norm is calculated as Equation (8):

$$\text{Score}(c) = \sum_{i=1}^{C_{in}} \sum_{u=1}^{K} \sum_{v=1}^{K} |W_{c,i,u,v}| \tag{8}$$

Where c is the channel index. Finally, retain the first C' channels with the highest scores, set the weights of the remaining channels to 0, and complete the pruning of the entire group.

3.3.2 Model Evaluation and Reward Calculation

Fine-tune the pruned model, and calculate the accuracy $\text{acc}_{current}$, $\text{FLOPs}_{current}$, and $\text{Params}_{current}$ on the validation set. Then, calculate the reward value R according to the reward function.

Fine- tune usually uses a small amount of training data for training over several epochs to restore the performance of the pruned model. When calculating the reward, to reduce reward variance, a baseline reward b is introduced and calculated using an exponential moving average: $b \leftarrow \alpha_b \cdot b + (1 - \alpha_b) \cdot R$, where α_b is a smoothing coefficient (e.g., 0.99). By comparing with the baseline, the superiority or inferiority of the current pruning strategy can be more accurately evaluated, accelerating the convergence of training.

3.3.3 Experience Storage and Training

Store the quadruplet (s_l, a_l, R, s_{l+1}) into the experience replay buffer. When the amount of data in the buffer reaches M , randomly sample data with a batch size of N for training. The loss function of the critic network is Equation (9):

$$L_Q = \frac{1}{N} \sum_{i=1}^{N} (y_i - \min(Q_1(s_i, a_i), Q_2(s_i, a_i)))^2 \tag{9}$$

This formula enables the critic network to learn to accurately estimate the state-action pair value by minimizing the mean squared error. Among them, the target value is $y_i = r_i - b + \gamma \min(Q_1^{'}(s_{i+1}, \pi'(s_{i+1})), Q_2^{'}(s_{i+1}, \pi'(s_{i+1})))$,where r_i is the sampled reward, b is the baseline reward, and γ is the discount factor, which is used to balance short - term and long - term rewards. $Q_1^{'}$, $Q_2^{'}$, and π' are target networks.

Update the actor network every K rounds, by minimizing the loss function is Equation (10):

$$L_\pi = -\frac{1}{N} \sum_{i=1}^{N} \min(Q_1(s_i, \pi(s_i)), Q_2(s_i, \pi(s_i))) \tag{10}$$

The parameters of the actor network are updated using the gradient descent method. The negative sign converts the problem of maximizing the Q-value into the problem of minimizing the loss. By optimizing this loss, the actor network learns to generate pruning strategies that can obtain high rewards, and gradually improves the pruning effect.

4. Experimental Analysis

4.1 Dataset

The experiment selects the CIFAR-100 dataset, which was released by the Canadian Institute for Advanced Research (CIFAR) in 1998. This dataset contains 100 classes, with 600 color images in each class. Among them, 500 images are used for training and 100 for testing, totaling 60,000 images. All

ISSN 2616-5775 Vol. 8, Issue 7: 79-86, DOI: 10.25236/AJCIS.2025.080710

images have a size of 32×32 pixels.

We employed two classic deep convolutional neural network models, ResNet56 and VGG19, for the experiments. ResNet56 possesses powerful feature extraction capabilities and demonstrates excellent performance in image classification tasks. VGG19 is renowned for its uniform network architecture and large receptive field, enabling it to extract rich image features.

4.2 Experimental results

In the experiment, three criteria were adopted to evaluate the compression model: test Top-1 accuracy,FLOPs compression ratio $= 1 - FLOPs_{current} / FLOPs_{origin}$, and parameter count compression ratio $= 1 - Params_{current} / Params_{origin}$. Additionally, comparisons were made with two other structured pruning techniques, namely DepGraph [10] and GReg [20]. We evaluated their performance at different sparsity levels using two channel sparsity ratios (20% and 50%).The specific experimental analysis is shown in Table 1.

Table 1: Comparison with other state-of-the-art methods on the CIFAR-100 dataset

Model (Accuracy,FlOPs,Para. Num.)	Method	Pruned Accuracy/FLOPs Ratio/Para. Num. Ratio	
		20%	50%
ResNet56 (72.62,128.63M,0.86M)	Greg	70.75/49.23/36.15	63.21/79.63/77.41
	DepGraph	71.38/50.95/39.25	64.55/82.01/78.36
	Ours	71.49/51.05/37.28	64.57/81.53/79.56
VGG19 (73.28,419.53M,39.35M)	Greg	72.95/28.45/35.46	71.55/52.72/71.31
	DepGraph	73.21/29.32/35.78	72.69/43.97/71.64
	Ours	73.06/24.53/36.01	72.63/52.77/71.66

5. Conclusion

In this study, we proposed a structured pruning method based on reinforcement learning, aiming to optimize the neural network structure to improve computational efficiency and model performance. Experiments demonstrate that our method effectively reduces the number of model parameters and computational complexity while maintaining high classification accuracy when applied to ResNet56 and VGG19 models on the CIFAR-100 dataset. Compared with traditional methods, the reinforcement learning strategy in our method dynamically adjusts the pruning ratio, which not only removes redundant parameters but also retains the key feature representation capabilities, showing good generalization. Future work will focus on exploring the application of this method on larger-scale datasets and more complex network architectures.

References

[1] Simonyan K ,Zisserman A .Very Deep Convolutional Networks for Large-Scale Image Recognition. [J].CoRR,2014,abs/1409.1556

[2] Chowdhery A, Narang S, Devlin J, et al. Palm: Scaling language modeling with pathways[J]. Journal of Machine Learning Research, 2023, 24(240): 1-113.

[3] Hongbing Z .Application Research of Speech Signal Processing Technology Based on Cloud Computing Platform[J].International Journal of Information Technologies and Systems Approach (IJITSA),2021,14(2):20-37.

[4] You H, Li C, Xu P, et al. Drawing early-bird tickets: Towards more efficient training of deep networks.[J].CoRR,2019,abs/1909.11957

[5] Han S, Mao H ,Dally J W .Deep Compression: Compressing Deep Neural Network with Pruning, Trained Quantization and Huffman Coding.[J].CoRR,2015,abs/1510.00149

[6] Gao S, Zhang Y, Huang F, et al. BilevelPruning: unified dynamic and static channel pruning for convolutional neural networks[C]//Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. 2024: 16090-16100.

[7] Lin H, Xu H, Wu Y, et al. Duquant: Distributing outliers via dual transformation makes stronger quantized llms[J]. Advances in Neural Information Processing Systems, 2024, 37: 87766-87800.

[8] Lv J, Yang H, Li P. Wasserstein distance rivals kullback-leibler divergence for knowledge distillation[J]. Advances in Neural Information Processing Systems, 2024, 37: 65445-65475.

[9] Guo Z, Cheng X, Wu Y, et al. A wander through the multimodal landscape: Efficient transfer learning via low-rank sequence multimodal adapter[C]//Proceedings of the AAAI Conference on Artificial

ISSN 2616-5775 Vol. 8, Issue 7: 79-86, DOI: 10.25236/AJCIS.2025.080710

Intelligence. 2025, 39(16): 16996-17004.
[10] Fang G, Ma X, Mi M B, et al. Isomorphic pruning for vision models[C]//European Conference on Computer Vision. Cham: Springer Nature Switzerland, 2024: 232-250.
[11] Liu J, Tang D, Huang Y, et al. Updp: A unified progressive depth pruner for cnn and vision transformer[C]//Proceedings of the AAAI Conference on Artificial Intelligence. 2024, 38(12): 13891-13899.
[12] Wang B, Kindratenko V. Rl-pruner: Structured pruning using reinforcement learning for cnn compression and acceleration[J]. arXiv preprint arXiv:2411.06463, 2024.
[13] Wang Z, Li C, Wang X. Convolutional neural network pruning with structural redundancy reduction[C]//Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. 2021: 14913-14922.
[14] Benbaki R, Chen W, Meng X, et al. Fast as chita: Neural network pruning with combinatorial optimization[C]//International Conference on Machine Learning. PMLR, 2023: 2031-2049.
[15] Diao E , Wang G , Zhan J ,et al.Pruning Deep Neural Networks from a Sparsity Perspective[J]. ArXiv, 2023, abs/2302.05601.DOI:10.48550/arXiv.2302.05601.
[16] Mnih V, Kavukcuoglu K, Silver D, et al. Human-level control through deep reinforcement learning [J]. nature, 2015, 518(7540): 529-533.
[17] Mnih V, Badia A P, Mirza M, et al. Asynchronous methods for deep reinforcement learning[C]//International conference on machine learning. PmLR, 2016: 1928-1937.
[18] Lillicrap P T ,Hunt J J ,Pritzel A , et al.Continuous control with deep reinforcement learning. [J]. CoRR,2015,abs/1509.02971
[19] Fujimoto S, Hoof H, Meger D. Addressing function approximation error in actor-critic methods[C]//International conference on machine learning. PMLR, 2018: 1587-1596.
[20] Wang H , Qin C , Zhang Y ,et al.Neural Pruning via Growing Regularization[J]. 2020. DOI:10.48550/arXiv.2012.09243.