mathematics

MDPI

Article

Automatic Compression of Neural Network with Deep Reinforcement Learning Based on Proximal Gradient Method

Mingyi Wang [1,2], Jianhao Tang [1,2], Haoli Zhao [1,3,*], Zhenni Li [1,2] and Shengli Xie [4,5]

1 School of Automation, Guangdong University of Technology, Guangzhou 510006, China
2 Guangdong-Hong Kong-Macao Joint Laboratory for Smart Discrete Manufacturing, Guangzhou 510006, China
3 111 Center for Intelligent Batch Manufacturing Based on IoT Technology (GDUT), Guangzhou 510006, China
4 Key Laboratory of Intelligent Detection and The Internet of Things in Manufacturing, Guangzhou 510006, China
5 Guangdong Key Laboratory of IoT Information Technology, Guangzhou 510006, China
* Correspondence: zhaohli@gdut.edu.cn

Abstract: In recent years, the model compression technique is very effective for deep neural network compression. However, many existing model compression methods rely heavily on human experience to explore a compression strategy between network structure, speed, and accuracy, which is usually suboptimal and time-consuming. In this paper, we propose a framework for automatically compressing models through the actor–critic structured deep reinforcement learning (DRL) which interacts with each layer in the neural network, where the actor network determines the compression strategy and the critic network ensures the decision accuracy of the actor network through predicted values, thus improving the compression quality of the network. To enhance the prediction performance of the critic network, we impose the L_1 norm regularizer on the weights of the critic network to obtain a distinct activation output feature on the representation, thus enhancing the prediction accuracy of the critic network. Moreover, to improve the decision performance of the actor network, we impose the L_1 norm regularizer on the weights of the actor network to improve the decision accuracy of the actor network by removing the redundant weights in the actor network. Furthermore, to improve the training efficiency, we use the proximal gradient method to optimize the weights of the actor network and the critic network, which can obtain an effective weight solution and thus improve the compression performance. In the experiment, in MNIST datasets, the proposed method has only a 0.2% loss of accuracy when compressing more than 70% of neurons. Similarly, in CIFAR-10 datasets, the proposed method compresses more than 60% of neurons, with only 7.1% accuracy loss, which is superior to other existing methods. In terms of efficiency, the proposed method also cost the lowest time among the existing methods.

Keywords: automatic compression; proximal gradient; network compression; structured pruning

MSC: 93C95

check for updates

Citation: Wang, M.; Tang, J.; Zhao, H.; Li, Z.; Xie, S. Automatic Compression of Neural Network with Deep Reinforcement Learning Based on Proximal Gradient Method. *Mathematics* **2023**, *11*, 338. https://doi.org/10.3390/math11020338

Academic Editor: Florin Leon

Received: 17 December 2022
Revised: 2 January 2023
Accepted: 4 January 2023
Published: 9 January 2023

1. Introduction

Deep neural networks (DNNs) are widely used in computer vision and machine learning, such as image recognition [1–3], action detection [4–6], target detection [7–9], or semantic segmentation [10–12]. To achieve better performances on these tasks, the design of DNNs becomes more and more complex in terms of depth and width, which hinders the application of DNNs in resource-constrained mobile environments or embedded devices. To overcome this problem, model compression emerges as a promising method to obtain a compact network.

Popular model compression techniques include quantization [13,14], which uses fewer bits to represent network weights, and low-rank approximation [15,16] focuses on dividing a large original matrix into several smaller matrices. Thereby, the computation of the original large matrix becomes the multiplication of several small matrices. Knowledge distillation [17–19] is to train a compact network based on the knowledge of a teacher network and pruning [20–22] is to develop a small and efficient neural network by removing redundant weights in the weight matrix. The pruned network can be well employed in mainstream hardware to obtain considerable speed-up, so pruning has attracted much attention.

There are two major branches of network pruning. One branch is unstructured pruning [23,24], which prunes at the level of individual weights. The other branch is structured pruning [25–28], which prunes at the level of neurons (or channels). Unstructured pruning usually reduces more weight than structured pruning [29]. However, it has the drawback that the resulting weight matrices are irregular distribution, which requires dedicated hardware for deployment. For structured pruning, it realizes network compression by pruning one row or one column in the weight matrix, so that the weight matrices are in regular distribution regular, and easy to deploy to practical applications and calculations.

Structured pruning is divided into the following three categories.

(1) One is the pruning method driven by regularization, for example, ref. [25] uses the group sparse lasso penalty to prune network neurons. However, the compression rate is dependent on the adjustment of hyperparameters, and the hyperparameters require analysis and adjustment empirically. Moreover, the layers in networks are correlated, which causes the compression rate of each layer will affect to the other layers. So, the compression rate is designed by adjusting the parameters without considering the relationship among layers of networks.

(2) The second is a pruning method driven by auxiliary parameters, such as AutoPrune [28], which takes a set of auxiliary parameters and optimally trains them to prune the network instead of the original weights, and reduces the dependence on the networks and the time for trial on hyperparameters. However, pruning the network through a set of auxiliary parameters, the corresponding hyperparameters still need to be adjusted, so manual empirical is still required.

(3) The third is a deep reinforcement learning (DRL)-based pruning method, such as AMC [30], which achieves network pruning by making decisions on the compression rate of the neural network through DRL without complex hyperparameter tuning. However, the redundant weights in the DRL network can lead to the problem of an inefficient pruning process.

Overall, the existing methods rely heavily on the prior knowledge of experts, which is not only suboptimal but also very time-consuming. As the layer in deep neural networks are interconnected with each other, and have different sensitivities to compression rates, adjusting the parameters to design the compression rate will ignore the integrity of the neural networks. This will affect compression performance. Although DRL methods can effectively alleviate these problems, the redundant weights in DRL still affect the compression efficiency. Therefore, it is necessary to implement an automatic and efficient model compression.

In this paper, we propose a new automatic pruning method based on deep reinforcement learning (DRL). Specifically, we employ DRL to interact with each layer in the network, and adaptively determine the compression rate of each layer in the network. In the compression process, the relationship between the various layers of the network is taken into account, which improves the performance of the compression model. In DRL, there are the critic network and the actor network, where the actor network is the decision network and the critic network assists the actor network in making more accurate decisions by predicting the value. We imposed the L_1 norm regularizer on the weights of the actor network and the critic network and optimized their weights using the proximal gradient method to remove the redundant weights. As a result, to remove the redundant weights which affect both efficiency and representation accuracy of the network. Therefore, the compact DRL can provide more accurate pruning decisions efficiently.

As shown in Figure 1, we implement the proposed method by modeling the pruning process as a Markov decision process. Specifically, within a training batch, the agent obtains the state information S_t of the L_t layer of the neural network from the pruning environment, and then outputs the pruning rate as the action a_t. The agent performs neuron compression on the current layer by a_t. Then, the reward is obtained and returned to the agent by validating on the validation set. Similarly, the agent interacts with the next layer L_{t+1} of the neural network to obtain the state information S_{t+1} of the next layer and the reward. The training batch ends when the agent has finished interacting with the last layer of the network. If the DRL has converged, the training is finished, otherwise, the training continues in the next training batch. To demonstrate the effectiveness of the proposed method, we completed experiments on two network models (LeNet-300-100 (784-300-100-10), MLP (3072-1024-1024-10)) and three datasets (MNIST, Fashion MNIST, CIFAR-10). The results show that the proposed method can compress more than 70% of neurons and maintain good performance. Moreover, it takes less time to achieve higher compression rates. In short, the proposed method can obtain a better lightweight network through automatic and efficient means.

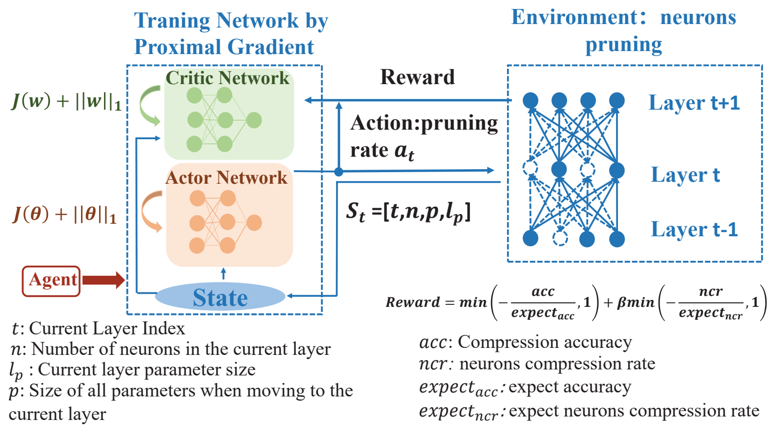
Traning Network by Proximal Gradient
Environment: neurons pruning
Critic Network
Actor Network
$J(w) + ||w||_1$
$J(\theta) + ||\theta||_1$
Agent
State
Reward
Action:pruning rate a_t
S_t =[t,n,p,l_p]
Layer t+1
Layer t
Layer t-1
$Reward = min\left(-\frac{acc}{expect_{acc}}, 1\right) + \beta min\left(-\frac{ncr}{expect_{ncr}}, 1\right)$
t: Current Layer Index
n: Number of neurons in the current layer
l_p : Current layer parameter size
p: Size of all parameters when moving to the current layer
acc: Compression accuracy
ncr: neurons compression rate
$expect_{acc}$: expect accuracy
$expect_{ncr}$: expect neurons compression rate

Figure 1. Neural network compression based on deep reinforcement learning (where the dotted line indicates the neurons that are pruned off and the corresponding weights).

We propose a new DRL-based automatic pruning method. In the actor–critic structure of DRL, the compression rate of each layer of the network is automatically determined by the actor network and the critic network evaluates the decision accuracy of the actor network by predicting the value. The contributions have the following three points:

(1) To improve the predictive performance of the critic network, we impose the L_1 norm regularizer on the weights of the critic network, so that the activation output can obtain distinct features in representations, thus improving the prediction accuracy of the critic network.

(2) To improve the decision performance of the actor network, we impose the L_1 norm regularizer on the weights of the actor network such that the insignificant weights converge to 0, which can reduce the redundant weights in the actor network and improve the decision accuracy of the actor network.

(3) To improve the training efficiency and automatic compression performance of DRL, the proximal gradient method is employed to optimize the objective function of DRL by updating the weight parameters of the critic network and the actor network. A DRL-based automatic compression algorithm was obtained by the proximal gradient optimization method to achieve automatic compression of the network.

In the following, we first discuss the related work in Section 2, and then introduce our method in detail in Section 3. The experimental results of our method are provided and analyzed in Section 4. Finally, conclusions are given in Section 5.

2. Related Work

Network pruning is mainly divided into unstructured pruning and structured pruning. From a broader perspective, such as parameter quantization and low-rank factorization can be combined with network pruning to achieve higher compression and speed-up. Below we briefly discuss some related works.

Unstructured pruning removes the individual weight in the neural networks. Regression via the lasso [23] uses lasso as a regular term to punish unimportant weights in the network. Regularization with a non-convex penalty [24] uses a non-convex function to replace the usual convex function as a regular term. Although these methods achieve good pruning results, unstructured pruning causes an irregular distribution of weights and requires customized hardware to support the practical speed-up.

In contrast, structured pruning is especially advantageous by removing one row or one column in the weight matrix, which makes the weight distribution regular after pruning. It requires no additional operational processing for the application and is therefore easy to deploy on the application.

To this end, a regularization-driven pruning method is proposed, which reduces the model complexity by manually constraining the network to be retrained. Yoon et al. [26] used both exclusive sparse regularization based on $L_{(1,2)}$ norm and group sparse regularization based on $L_{(2,1)}$ norm on network weights, such that exclusive sparsity enforces the network weights to use input neurons that are as different as possible from the other weights. Louizos et al. [27] pruned the network by adding the L_0 norm as a regular term to the objective function, thereby encouraging the weight matrix to be exactly zero.

Auxiliary parameters drive pruning is also an effective means, which is not a direct regularization weight, but an auxiliary parameter for regularized aggregation gradient perturbation. In this way, the problem of inaccurate pruning due to instability and nonmost hyperparameters can be solved, which significantly improves the effectiveness of pruning. Xiao et al. [28], which is based on a set of auxiliary parameters to prune the neural network. During the training of the auxiliary parameters, the network weights are not affected by instability and noise, which makes the pruning process less affected by noise and hyperparameters. So, it reduces the time for trial and the dependence of the network on hyperparameters.

Deep reinforcement learning (DRL) driven pruning methods are used to improve the quality of model compression by efficiently sampling the design space with reinforcement learning, enabling automatic compression without any manual effort. He et al. [30] explore the compression rate of a neural network by training a deep reinforcement learning model and compressing the network by the compression rate. Its limitation is that it uses a traditional network pruning technique based on fixed regularization and only filter pruning is considered in the pruning process. As we will see later, the incompatibility between the adopted DRL framework and the problem to be solved affects the achievement of high pruning rates (the maximum reported pruning rate in (He et al. [30]) is only $5\times$ and is non-structured pruning).

Compared with regularization-driven pruning, the novelty of our method is twofold. First, the compression rate of each layer is automatically determined through DRL, without any manual effort. Second, the automatically decided compression rates consider the connection between the various layers of the network, which is better than the compression rate set by human experience. Compared with the pruning driven by auxiliary parameters, our novelty is to remove the dependence of the pruning process on the hyperparameters and improve the compression performance. Compared with DRL-based pruning methods, our novelty is the optimization of the DRL model. The redundant weights in DRL are removed and the weights are trained by the proximal gradient method to obtain the optimal weight solution. This improves the compression efficiency and performance.

3. Automatic Compression of Neural Network with Deep Reinforcement Learning Based on Proximal Gradient Method

Deep reinforcement learning (DRL) can automatically determine the compression rate of each layer with the state information of each layer in the neural network, and then perform pruning. For the training of the DRL, regularize the DRL using the L_1 norm, and using the proximal gradient method to update its parameters, which can obtain a more efficient weight parameters and improve the decision performance of the DRL. To promise compression efficiency, a reward that balances the compression rate and accuracy of the neural network is designed, which is used as feedback to the DRL after the networks are compressed. According to the reward, the DRL is constantly training decision performance. We model the network compression process as a Markov Decision Process, and determine the compression policy by implementing the Markov chain.

3.1. Markov Decision Problem Based Pruning Technology

Markov Decision Problem (MDP) is constructed by the interaction of the agent and the environment, and its elements include states, strategies, actions, and rewards. In MDP simulation, the agent perceives the current system state to obtain a policy and then implements actions on the environment according to the policy. Finally, the corresponding reward is obtained from the changed environment. The agent adjusts the actions by the rewards.

We redefine each element in MDP, which will be introduced separately below.

3.1.1. State

We express the state through a set of feature lists S = $[t,n,p,l_p]$. The features in this list express the information of a certain layer in the neural networks. Among them, t is the index of the current layer, n is the number of neurons in the current layer, p is the size of all parameters moved to the current layer, and l_p is the parameter size of the current layer. This set of feature lists is to allow the agent to distinguish the differences in information in each layer of the neural networks, so as to make the best decision for each layer.

3.1.2. Action

The action space is composed of actions, and each action corresponds to the sparsity a_t that determines how many neurons are pruned in the t-th layer. The pruning standard is to sort from small to large according to the L_1 norm size of each neuron in each layer, and then determine which neurons to be pruned by the action sparsity a_t. When a_t is taken for the layer, the pruned neuron is given by Equation (2).

$$Sort_n = Sort(||W_{n_t}||_1) \tag{1}$$

$$NeuronsPruned_t(a_t) = a_t Sort_n \tag{2}$$

where $||W_{n_t}||_1$ represents the L_1 norm of each neuron in the t-th layer, and $Sort_n$ represents the sorting result of the t-th layer.

3.1.3. Reward

Since the DRL agent learns the sparsity strategy with the goal of reward maximization, designing rewards can help the agent converge faster [31]. In order to balance the accuracy and sparsity of the network, we design the following reward function:

$$r(t) = min(-\frac{acc(t)}{expect_{acc}}, 1) + \beta min(-\frac{ncr(t)}{expect_{ncr}}, 1) \tag{3}$$

where *acc(t)* and *ncr(t)* represent the Validation accuracy and neuron compression rate at the moment of state t. $expect_{acc}$ and $expect_{ncr}$ represent the expected accuracy and neuron compression rate set by the user. β is the factor that balances the compression rate and accuracy of the network.

3.2. Pruning Based on DRL with the Proximal Gradient

In order to solve the Markov problem, we choose agents among many commonly DRL algorithms. Commonly used agents are Deep Q-Network (DQN) [32,33], which is a form of Q-learning [34,35]. Although the exploration is fast, it is not stable. Both Deep Deterministic Policy Gradient (DDPG) [30,36] and Proximal Policy Optimization (PPO) [37,38] are agents for solving continuous control problems, but the output of PPO is a probability distribution, while the output of DDPG is directly an action. The accuracy of the deep neural network is sensitive to the sparsity of each layer, resulting in an action space that should not be obtained too large [30]. Therefore, instead of considering the exploration on discrete spaces, we employ a DDPG, which is essentially a continuous control strategy, as an agent. The compression strategy is learned by trial and error: penalizing the loss of accuracy while driving the network to shrink. DDPG includes a value strategy network (critic) and a learning strategy network (actor). The learning strategy network is responsible for outputting actions, and the value strategy network is responsible for evaluating actions by output Q value. In addition, the action value and Q value of the next state are required. Therefore, two target networks (target actor and target critic) are used to obtain the action value and Q value of the next state.

The objective functions of the actor network and the critic network as follows:

$$LOSS = \frac{1}{N}\sum_i [(Q(S_i, \mu_{\theta^t}(S_i)|\theta^Q))] \tag{4}$$

$$\begin{aligned} LOSS \quad = \quad & \frac{1}{N}\sum_i [r_t + \gamma Q_{w^t{}_{targ}}(S_{i+1}, \mu_{\theta^t{}_{targ}}(S_{i+1})|\theta^Q) \\ & - Q_{w^t}(S_i, \mu_{\theta^t}(S_i)|\theta^Q)]^2 \end{aligned} \tag{5}$$

where $\mu_{\theta^t}(.)$ is the actions in the current state output by the actor, and $\mu_{\theta^t{}_{targ}}(.)$ is the action in the next state output by the target actor. $Q_{w^t{}_{targ}}(.)$ is Q value in the next state output by the target critic, and $Q_{w^t}(.)$ is Q value in the current state output by the critic.

Actor–critic (AC) [39,40] is an important learning framework in deep reinforcement learning. To improve the learning efficiency and convergence of deep reinforcement learning, AC learning control with regularization [41] proposed a learning-control method based on regularization and feature selection, which effectively improved the learning performance of deep reinforcement learning. Inspired by this, we propose to impose the L_1 norm regularizer on the weights of the actor network and the critic network, and employ the proximal gradient method to update the weight. The main idea of the proximal gradient method is to solve the sub-problem that is more efficient than the original problem by iteratively updating the proximal operator. The proximal gradient method includes the proximal gradient ascent method and the proximal gradient descent method. The proximal gradient method is usually used to optimize an objective function composed of a smooth term loss function and a non-smooth penalty function. The proximal gradient method for the regularization target first obtains the intermediate variable $W^{t+\frac{1}{2}}$ by gradient ascent or descent using the gradient obtained only based on the loss function calculation, and then optimizes the regularization term when performing the Euclidean projection on it to the solution space. So, Equations (4) and (5) are transformed as follows:

$$\min_{\theta^t} J(\theta^t) = \frac{1}{N}\sum_i [(Q(S_i, \mu_{\theta^t}(S_i)|\theta^Q))] + ||\theta^t||_1 \tag{6}$$

$$\min_{w^t} J(w^t) = \frac{1}{N}\sum_i [r_t + \gamma Q_{w^t{}_{targ}}(S_{i+1}, \mu_{\theta^t{}_{targ}}(S_{i+1})|\theta^Q) - Q_{w^t}(S_i, \mu_{\theta^t}(S_i)|\theta^Q)]^2 + ||w^t||_1 \quad (7)$$

The actor network minimizes the objective function to update its parameters θ^t, and similarly, the critic network minimizes the objective function to update its parameters w^t. Therefore, θ^t and w^t can be updated iteratively via proximal gradient ascent and proximal gradient descent:

$$\theta^{t+\frac{1}{2}} \leftarrow \theta^t + \alpha \nabla J(\theta^t) \quad (8)$$

$$w^{t+\frac{1}{2}} \leftarrow w^t - \delta \nabla J(w^t) \quad (9)$$

$$\theta^{t+1} \leftarrow sign(\theta^{t+\frac{1}{2}}) max(|\theta^{t+\frac{1}{2}}| - \lambda lr, 0) \quad (10)$$

$$w^{t+1} \leftarrow sign(w^{t+\frac{1}{2}}) max(|w^{t+\frac{1}{2}}| - \lambda lr, 0) \quad (11)$$

here, λ, α, and δ is coefficient. lr is the learning rate. $\theta^{t+\frac{1}{2}}$ and $w^{t+\frac{1}{2}}$ is the intermediate variable.

In addition, the parameters update of the target network is as follows:

$$\theta^t_{targ} \leftarrow \tau\theta^t + (1-\tau)\theta^t_{targ} \quad (12)$$

$$w^t_{targ} \leftarrow \tau w^t + (1-\tau) w^t_{targ} \quad (13)$$

here, τ is the learning rate.

3.3. Proposed Algorithom

For the deep reinforcement learning agent to accurately determine the compression strategy of the neural network, the agent needs to be trained extensively. In the beginning, the agent conducts a preliminary exploration of the network and obtains a large amount of training data. We have an experience pool, which is used to store the training data that the agent explores in the early stage. When the data reaches a certain amount, the agent will use the training data for learning and training, and constantly update the parameters of the actor network and the critic network. The agent also keeps learning and training in the exploration and repeats it until the training converges. For a detailed description, please refer to Algorithm 1 and Figure 2.

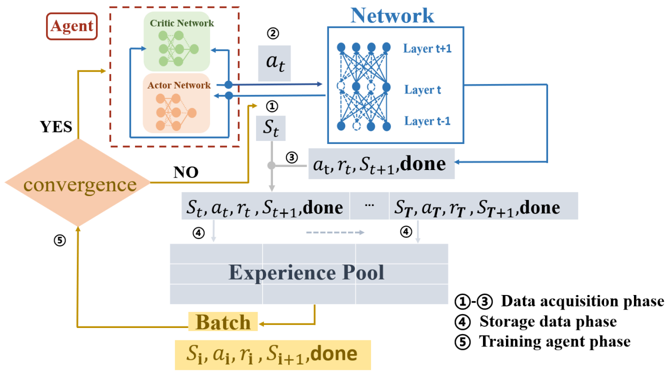

Figure 2. Training process of deep reinforcement learning compressed neural network.

Algorithm 1 Neural network pruning algorithm by deep reinforcement learning

```
Initialization:
    $t$ : current layer
    $s_t$: current state
    $a_t$: pruning rate be created by Actor
    $s_{t+1}$: next state
    $r_t$: reward of pruning one layer
    $buffer_{size}$: desired size of data to be collected
    $replay$: experience pool
    $replay_{size}$: experience pool size
Pruning:
 1: done = true
 2: while done do
 3:     $t = 0$
 4:     $a_t \leftarrow$ Actor ($s_t$)
 5:     Pruning $t$ layer by (1)(2) with $a_t$
 6:     $r_t \leftarrow$ Verify network acc and ncr
 7:     $replay \leftarrow [s_t, a_t, r_t, s_{t+1}, done]$
 8:     if ($t$+1) is last layer then
 9:         done = false
10:         if $replay_{size} < buffer_{size}$ then
11:             continue
12:         end if
13:     else
14:         $t = t+1$
15:     end if
16:     for batch in $replay$ do
17:         use batch update actor and critic by the proximal gradient (8)–(11)
18:     end for
19:     $Actor_{targ} \leftarrow Actor$ by (12)
20:     $Critic_{targ} \leftarrow Critic$ by (13)
21:     Clear replay
22:     if all parameters is convergence then
23:         done = false
24:     else
25:         continue
26:     end if
27: end while
```

4. Experiments

All experiments in this paper are based on the python programming language for simulation. The operating system is Windows, and all algorithms are coded in pytorch 1.1.0. CUDA 10.0.

4.1. Experimental Setup

This paper mainly uses three public experiment datasets and two different neural networks.

4.1.1. MNIST Datasets

The dataset is composed of 70,000 handwritten digital grayscale images with a size of 28 × 28. Each category has 6000 training examples and 1000 test examples. There are 10 categories in total, representing numbers 0 to 9. For the basic network, this article uses the LeNet300-100 neural network, which has four layers, an input layer, an output layer, and two hidden layers. The input layer has 784 neurons, the two hidden layers have 300 neurons and 100 neurons, respectively, and the output layer has 10 neurons (784-300-100-10).

4.1.2. Fashion MNIST Datasets

There are 10 types of images in the Fashion MNIST dataset: t-shirts, jeans, pullovers, skirts, coats, sandals, shirts, sneakers, bags, and boots. The training dataset has 6000 samples of each of the above 10 types, while the test dataset has 1000 samples of each of the above 10 types. Thus, there are a total of 60,000 samples in the training dataset and a total of 10,000 samples in the test dataset. For the basic network, the LeNet300-100 neural network (784-300-100-10) is also used.

4.1.3. Cifar-10 Datasets

The dataset is composed of 60,000 pictures with a size of 32×32. The pictures have 10 animal and vehicle categories. Among them, 50,000 pictures are used for training and 10,000 pictures are used for testing. Therefore, for the basic network, this paper uses a custom neural network with four layers, an input layer, an output layer, and two hidden layers. The input layer has 3072 neurons, the two hidden layers have 1024 neurons, and the output layer has 10 neurons (3072-1024-1024-10).

4.2. Experimental Comparison and Evaluation Index

We implement the state-of-the-art algorithms, namely, Combined Group and Exclusive Sparsity (CGES) [26], AutoPrune [28], Automl for Model Compression (AMC) [30], and L_0 norm [27] to compare with our proposed algorithm Automatic Compression of Neural Network (ACNN). CGES regularizes the neurons in the network by $L_{(2,1)}$ norm, so that some neurons are eliminated. The L_0 norm regularizes the network through the L_0 norm. During the training process, the network is pruned by encouraging the weight of the neuron to be exactly zero. AutoPrune, which prunes the network with a set of optimized auxiliary parameters. It reduces the complexity of relying on a priori knowledge and reduces trial and error time. AMC interacts with the network through deep reinforcement learning to decide the compression rate of each layer of the network and achieves network compression based on the compression rate.

In order to better verify the effect of the proposed method, we also compared the proposed method without the proximal operator. There are three evaluation indicators for comparison: accuracy loss (Error), neurons compression rate (NCR), and running time (time). The accuracy loss represents the recognition performance of the neural network, which is defined by the loss of the sparse network accuracy relative to the original network accuracy. The neurons compression rate represents the ratio of the total number of original neurons in the network to the total number of neurons remaining in the network after compressed, and the efficiency represents the time spent by different compression methods. Other indicators in the experiment include: pruned layer neurons distribution (layers), the network sparsity of DRL (DRL spa), and the floating point computation degree of the compressed network (flops).

4.3. Convergence of Deep Reinforcement Learning in Pruning

Figure 3 show the learning and training in the process of deep reinforcement learning exploration and decision-making under different datasets. When DRL explores, we set the learning rate to 0.001, and because the complexity of each dataset is different, the iteration cycle we design is also different. As shown in the figure, as the number of iterations increases, the loss of the deep reinforcement learning network training continues to converge, and finally stabilizes.

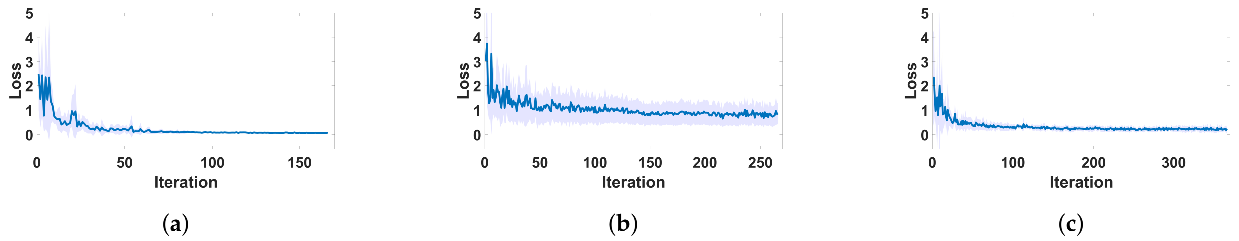

Figure 3. The above shows the training loss of DRL on MNIST (**a**), Fashion MNIST (**b**), and Cifar-10 (**c**), respectively.

4.4. Balance Factor of the Reward

From Equation (5), it is clear that the training of the critic network is positively correlated with the reward r. Therefore, the performance of the neural network after automatic compression by DRL is also positively correlated with reward r. For this reason, we set the reward R as shown in Equation (3), where β is used to balance the compression rate and accuracy of the compressed neural network. To explore better automatic compression results, we adjust the β as shown in Figure 4. As the β increases, the compression rate of the neural network keeps increasing, and conversely, the corresponding accuracy keeps decreasing. Therefore, the best balance of compression rate and accuracy can be obtained by adjusting the β to make the best result after automatic compression.

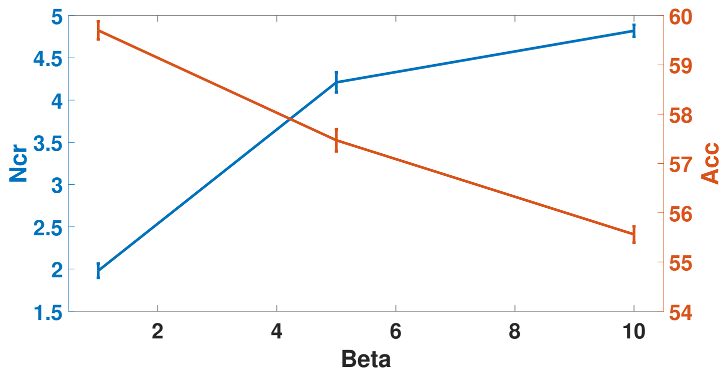

Figure 4. Network performance with beta after auto-compression.

4.5. Convergence of Reward

Figure 5 show the reward convergence of DRL during exploration and training with different datasets. As shown in the figures, the rewards obtained by DRL gradually converge after about 100 iterations of training. This indicates that DRL is able to obtain a stable compressed model after training with a large amount of training data.

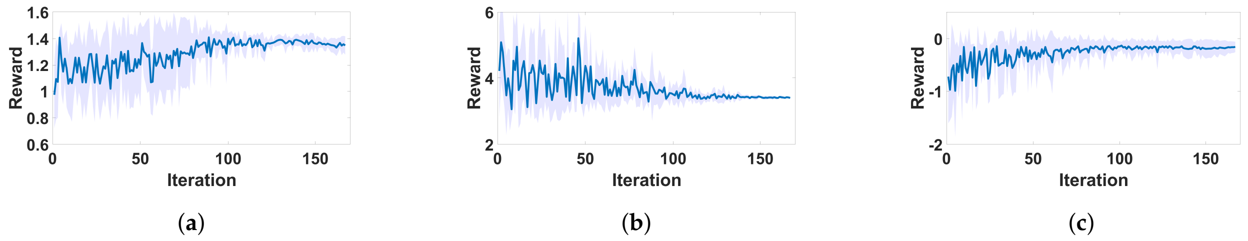

Figure 5. The above shows the reward of DRL on MNIST (**a**), Fashion MNIST (**b**), and Cifar-10 (**c**), respectively.

4.6. Results and Analysis

4.6.1. Pruning Performance on Mnist

As shown in Table 1 and Figure 6, the performance of our method is generally better than the other four methods. Compared with the other four methods, the neuron compression rate of CGES is 2.87, and the accuracy loss is 0.36%. The neuron compression rate of the L_0 norm is 3.06, and the accuracy loss is 0.23%. The neuron compression rate of AutoPrune is 3.23, and the accuracy loss is 0.22%. The neuron compression rate of AMC is 3.3 and the accuracy loss is 0.3. Our method can reach the highest neuron compression rate of 3.46, which compresses 16 neurons more than AMC, and the accuracy loss is only 0.22%. For the floating point computation degree, the proposed method in this paper also requires the lowest floating point computation degree of only 28% while achieving the lowest accuracy loss. The results show that the proposed method in this paper requires less computational resources while improving the compression performance.

Table 1. Comparison of Different Neuron Pruning Techniques in MNIST. (The best results are shown in bold).

Methods	Error (%)	Layers	NCR	FLOPS (%)
CGES [26]	0.36	221-97-94	2.87	35
L0 norm [27]	0.23	266-88-33	3.06	33
AutoPrune [28]	0.22	244-85-37	3.23	31
AMC [30]	0.30	225-88-45	3.30	30
ACNN	**0.20**	**219-81-43**	**3.46**	**28**

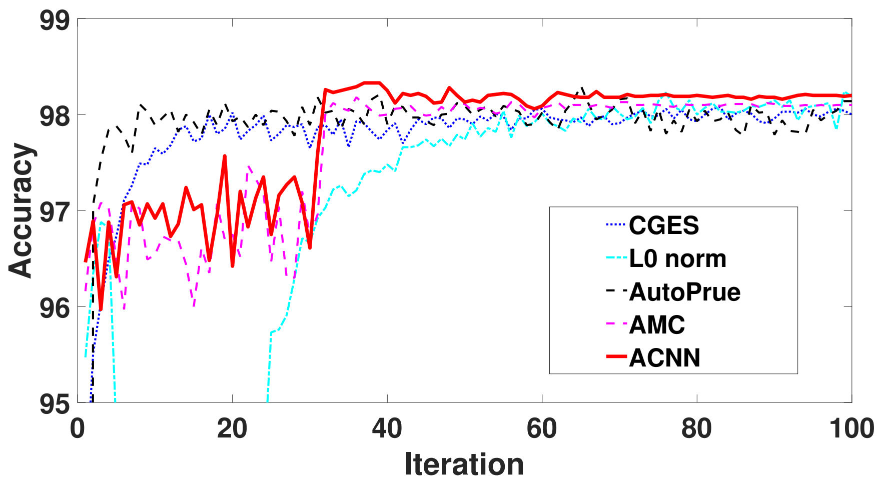

Figure 6. Accuracy comparison of each method under the MNIST dataset.

The method proposed in this paper takes the least amount of time. For non-DRL methods, the method proposed in this paper automatically explores the compression strategy using DRL and adaptively decides the compression rate of each layer in the neural network, which can effectively improve the compression efficiency of the network. For the DRL method, the proposed method optimizes the DRL. The redundant weights of the policy network in DRL are removed, which improves the decision efficiency of DRL and shortens the decision time. These can be see in Figure 7.

As shown in Table 2, the proposed method in this paper has an advantage over the AMC method in all aspects of performance. The main reason is that the method in this paper optimizes the actor network and critic network in DRL using proximal operation, which makes the decision making of the actor network more efficient and the prediction of the critic network more accurate by sparse of the network. In addition, the sparsity of actor network and critic network reaches 50%, which also reduces the computational memory of DRL in the pruning process and reduces the time needed to be spent on pruning, and improves the pruning efficiency.

Table 2. Performance comparison with unoptimized DRL compression model on MNIST. (The best results are shown in bold).

Methods	Error (%)	NCR	Time (s)	DRL SPA (%)
AMC [30]	0.30 ± 0.12	3.30 ± 0.20	1835.00 ± 15.72	0.00 ± 0.00
ACNN	**0.20** ± **0.16**	**3.46** ± **0.14**	**455.00** ± **2.88**	**96.00** ± **0.74**

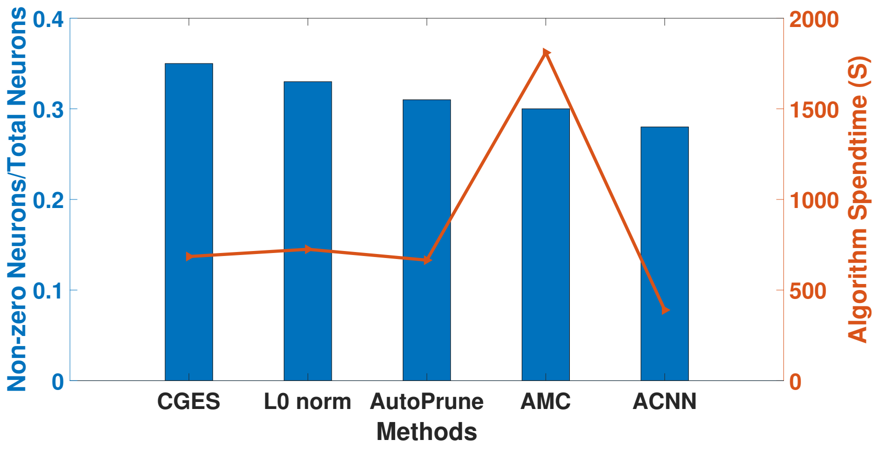

Figure 7. Network density and time spent on pruning in MNIST.

4.6.2. Pruning Performance on Fashion MNIST

For the Fashion MNIST dataset, as the dataset complexity scale becomes larger, the accuracy loss of the compressed model will be greater. As shown in Table 3 and Figure 8, among these methods, the highest accuracy loss is as high as 13.5%, but our method can obtain the lowest accuracy loss of 2.22%. This proves that the connection between the layers is taken into account in the compression process, which can reduce the large accuracy loss caused by compression.

Table 3. Comparison of different neuron pruning techniques in Fashion MNIST. (The best results are shown in bold).

Methods	Error (%)	Layers	NCR	FLOPS (%)
CGES [26]	13.50	702-15-7	1.64	60
L0 norm [27]	6.25	295-80-25	2.96	34
AutoPrune [28]	4.33	285-85-19	3.04	32
AMC [30]	2.40	141-81-42	4.48	23
ACNN	**2.22**	**185-35-20**	**4.92**	**20**

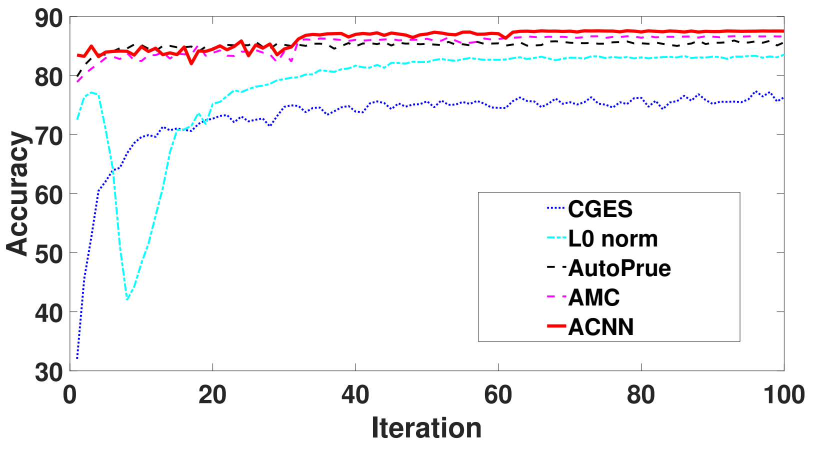

Figure 8. Accuracy comparison of each method under the Fashion MNIST dataset.

When the network compresses neurons that exceed 70% of the original network, the loss of accuracy only increases by 2.22%. For the other four methods, the model compression rate after using them are 1.64, 2.96, 3.04, and 4.48, respectively. The method proposed in this paper can reach the highest neuron compression rate of 3.75, which compresses 75 neurons more than AutoPrune, and the accuracy loss is only 2.22%. This is because the reward we designed balances the accuracy of the compression model with the compression rate, so that a compression model with both accuracy and compression rate can be explored.

In terms of efficiency, our method is nearly five times better than AutoPrune, and it takes less time than other methods when the compression ratio reaches the highest. This is due to the adaptive decision of the compression rate through deep reinforcement learning, removing artificial prior knowledge, shortening the trial and error time, and improving the compression quality. For the floating point computational degree, the computational complexity required by the method proposed in this paper is lower than all the other four methods by only 20%. This is mainly due to the sparse optimization of the actor network in DRL by the method in this paper, which makes the decision performance of the actor network improved and enables a higher degree of neural compression. These can be seen in Figure 9.

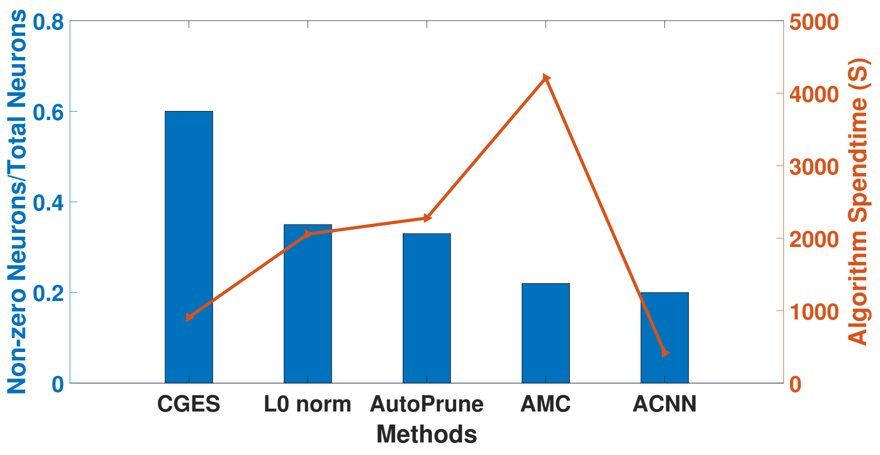

Figure 9. Network density and time spent on pruning in Fashion MNIST.

As shown in Table 4. Compared with AMC, the proposed method in this paper, after optimizing the actor network and critic network in DRL has only 2.22% accuracy loss and is able to achieve a compression rate of neurons of 4.92. This is mainly due to the improved decision making ability of the actor network on the compression rate of the layers in the neural network and the prediction of the compression rate by the critic network after using the proximal operation on DRL The DRL can make better decisions. Moreover, we design a reward that can balance the compression rate and accuracy, so that the compressed network model can have both a high compression rate and high prediction accuracy. In addition, the method proposed in this paper takes only one-tenth of the time of AMC, because the redundant weights in DRL are removed, reducing unnecessary computation time.

Table 4. Performance comparison with unoptimized DRL compression model on Fashion MNIST. (The best results are shown in bold).

Methods	Error (%)	NCR	Time (s)	DRL SPA (%)
AMC [30]	2.40 ± 0.10	4.48 ± 0.05	4210.00 ± 11.40	0.00 ± 0.00
ACNN	**2.22** ± **0.08**	**4.92** ± **0.04**	**544.00** ± **4.15**	**97.00** ± **0.50**

4.6.3. Pruning Performance on Cifar-10

In order to verify the performance of the sparse neural network on more complex classification tasks, we introduce the Cifar-10 dataset for experiments. The Cifar-10 dataset is an image with three channels, so we customize a deep neural network (3072-1024-1024-10) for compression experiments. In this experiment, the Cifar-10 dataset has the highest complexity, so the accuracy loss after network compression is greater than the above two datasets.

As shown in Table 5 and Figure 10, after using CGES, L_0, AutoPrune, and AMC methods to compress the network, the accuracy loss reached 28.7%, 26.04%, 24.04%, 7.63%, and the neuron compression rate was 1.61, 3.23, 3.6, 3.55, respectively. For the method proposed in this paper, the accuracy loss can be reduced to 7.1%, and the accuracy performance is significantly improved, and the neuron compression rate is higher than the other four methods, reaching 3.77, which is equivalent to 57 more neurons compressed than AutoPrune. Our method automatically explores the sparse structure of the network through deep reinforcement learning takes into account the integrity of the network, and improves the quality of the compression model. Therefore, for more complex recognition tasks, the compression model still has obvious performance advantages.

Table 5. Comparison of different neuron pruning techniques in Cifar-10. (The best results are shown in bold).

Methods	Error (%)	Layers	NCR	FLOPS (%)
CGES [26]	28.70	3035-142-3	1.61	62
L0 norm [27]	26.04	1036-260-290	3.23	31
AutoPrune [28]	24.04	900-240-282	3.60	28
AMC [30]	7.63	890-317-235	3.55	27
ACNN	**7.10**	**857-256-245**	**3.77**	**25**

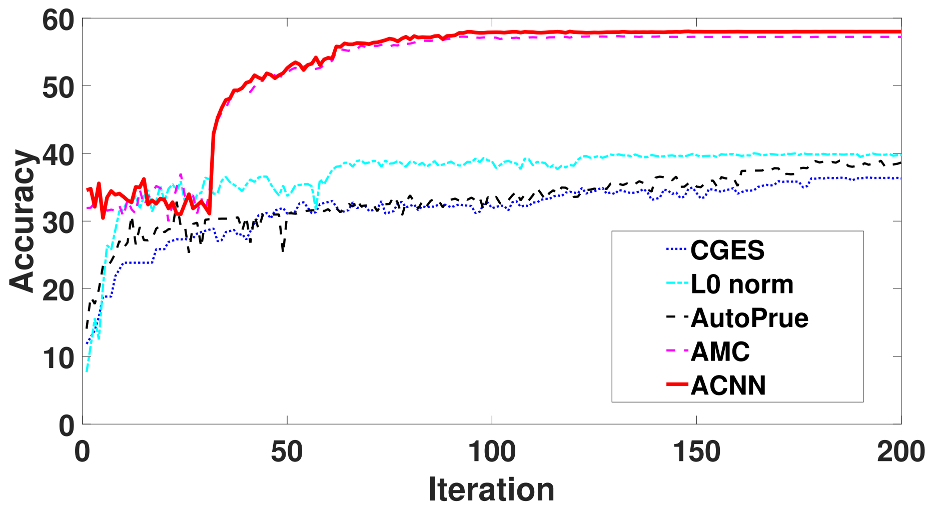

Figure 10. Accuracy comparison of each method under the Cifar-10 dataset.

The time efficiency of the compression process also has been improved by nearly 6 times that can be seen in Figure 11. As shown in Table 6, there is a significant advantage in both accuracy loss, neuron compression rate or time consumed after optimization of DRL.

The above verification results show that our method can find a sparse network architecture with balanced performance and efficiency on simple and complex datasets.

Table 6. Performance comparison with unoptimized DRL compression model on Cifar-10. (The best results are shown in bold).

Methods	Error (%)	NCR	Time (s)	DRL SPA (%)
AMC [30]	7.63 ± 0.65	3.55 ± 0.25	4157.00 ± 11.30	0.00 ± 0.00
ACNN	**7.10 ± 0.26**	**3.77 ± 0.21**	**756.00 ± 5.94**	**97.00 ± 0.60**

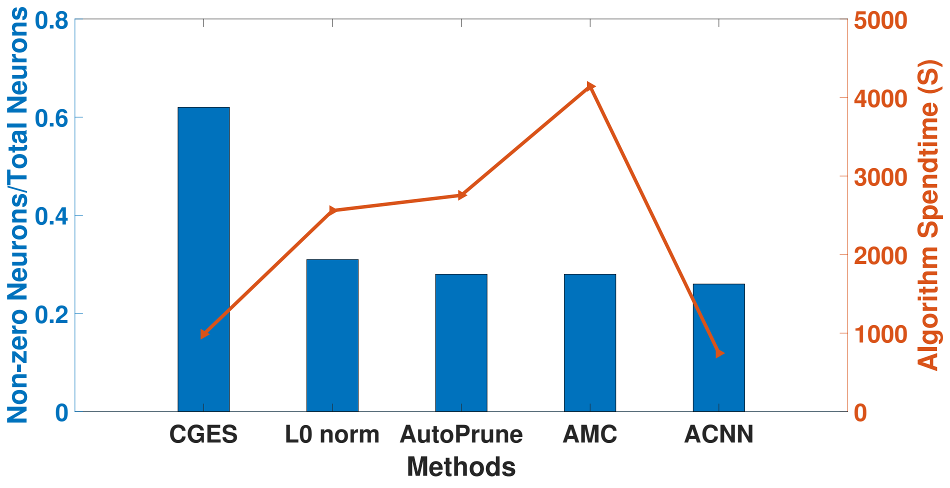

Figure 11. Network density and time spent on pruning in Cifar-10 dataset.

4.7. Ablation Study

As shown in Table 7, ablation experiments were conducted in this paper for optimized DRL with and without proximal gradient. From the table, it can be obtained that the compression performance of the network is improved in terms of compression accuracy and compression rate after using the proximal gradient for DRL optimization. This is because the actor networks and the critic networks in DRL can train better weight solutions after using proximal gradient, which improves their decision performance and prediction performance, resulting in a better structure of the network after compression by DRL.

Table 7. Performance comparison between using proximal and not using proximal with different datasets. (The best results are shown in bold).

Dateset	Method	Error (%)	Ncr
MNIST	ACNN without pro	0.21 ± 0.08	3.25 ± 0.07
	ACNN	**0.20 ± 0.16**	**3.46 ± 0.14**
Fashion MNIST	ACNN without pro	2.75 ± 0.62	3.62 ± 0.22
	ACNN	**2.22 ± 0.08**	**4.92 ± 0.04**
Cifar-10	ACNN without pro	7.71 ± 0.35	3.60 ± 0.06
	ACNN	**7.10 ± 0.26**	**3.77 ± 0.21**

4.8. Weight Distribution

Figures 12 and 13 show the distribution of weights in the layers of the neural network before and after compression by ACNN. The shaded part of the figure indicates that the weights are non-zero and the non-shaded part indicates that the weights are zero. As shown in the figures, the neural network has a dense and redundant distribution of weights before compression. After compression using the method proposed in this paper, the weights in the network are highly compressed and distributed regularly. This shows that the optimization of DRL improves the decision performance of the actor and achieves high compression of the network.

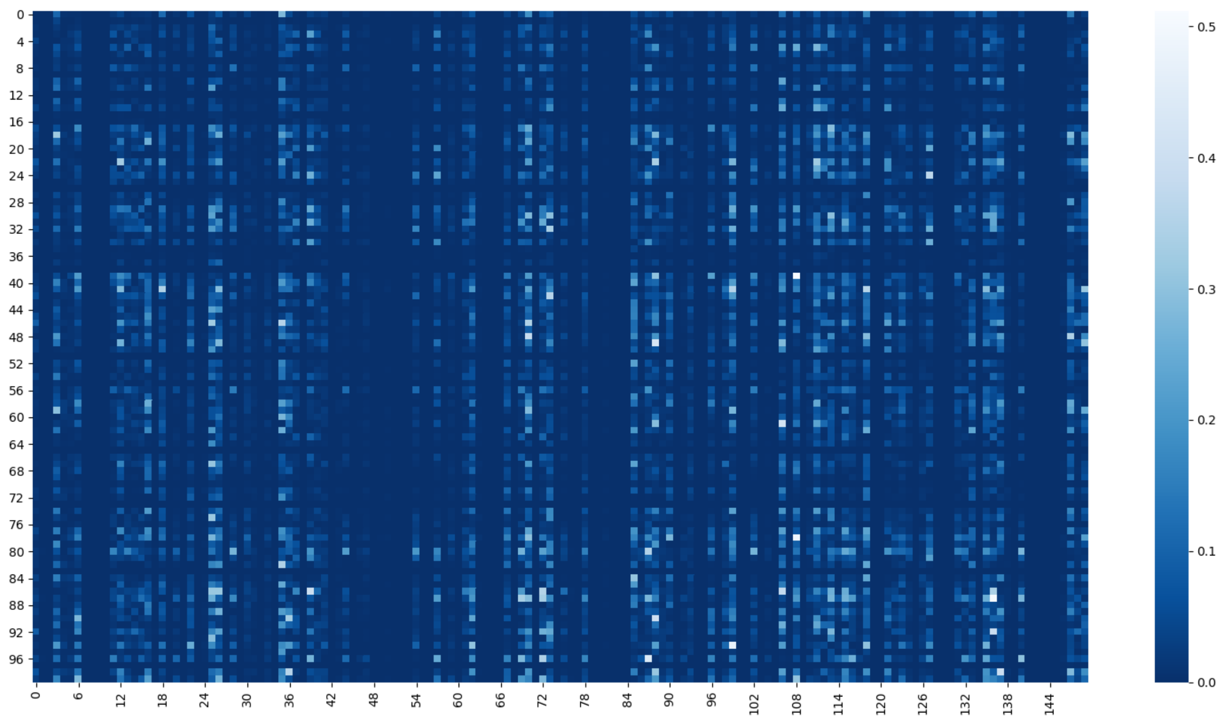

Figure 12. Distribution of neural network weights before compression by ACNN.

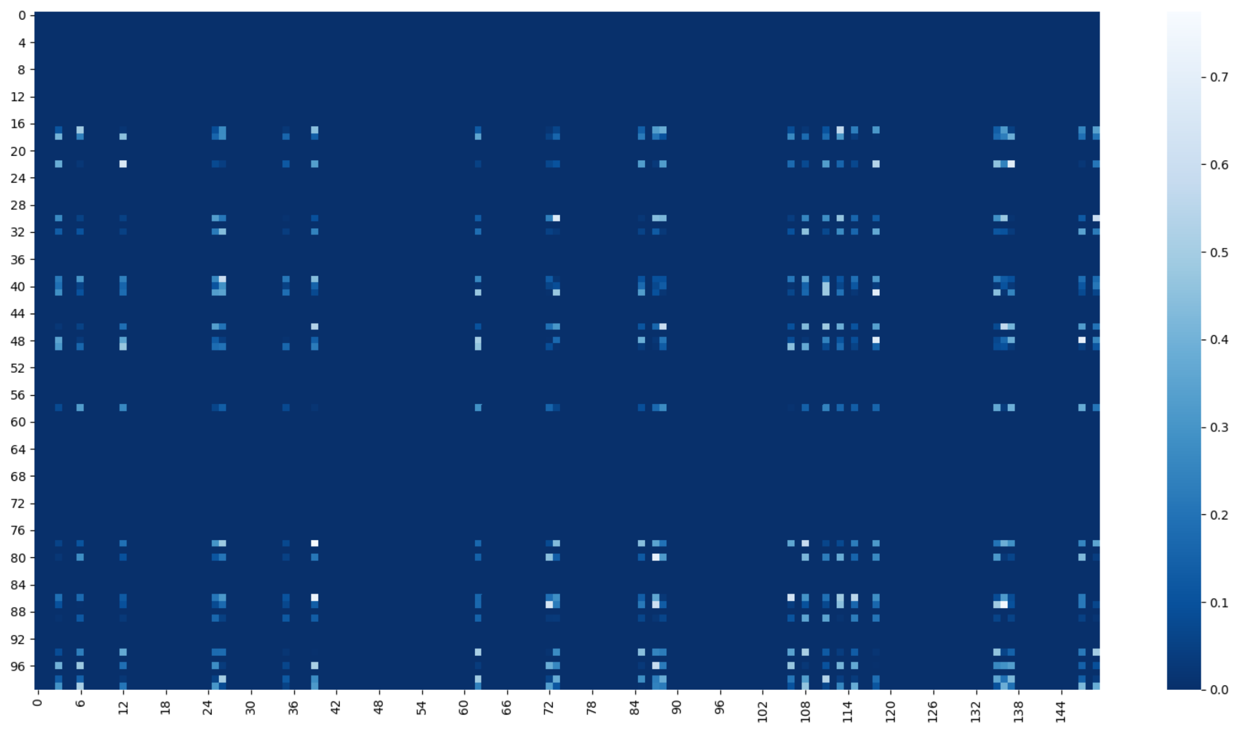

Figure 13. Distribution of neural network weights after compression by ACNN.

5. Conclusions

In this paper, we propose a new automatic pruning method based on proximal gradient. The purpose is to automatically compress the network by training an actor-critic structured DRL agent, which automatically decides the compression rate of each layer in the neural network by the actor network and guarantees the accurate decision of the actor network by the critic network prediction value. It not only improves the compression efficiency and quality but also removes the reliance on the human experience. To improve the prediction performance of the critic network in the agent, we impose the L_1 norm regularizer on the weights of the critic network to make the activation output can obtain distinct features in representations, thus enhancing the prediction accuracy of the critic network. In addition, to improve the decision performance of the actor network in the agent, we impose the L_1 norm regularizer on the weights of the actor network to enhance the decision accuracy of the actor network by retaining only the significant weights in the actor network. To further improve the training efficiency, we use the proximal gradient method to update the weights of the actor network and the critic network, which can obtain an effective distribution of weights and, thus, improve the compression performance. We show the performance comparison of the optimized agent and the unoptimized agent on the compression task, which demonstrates that the optimized agent improves all aspects of compression performance. Moreover, we also show ablation experiments using the proximal gradient method to update the DRL network weights. The results show that updating the weights using the proximal gradient method can obtain a better DRL compression agent and improve the performance of the network after compression. Compelling results have been demonstrated for LeNet300-100, and MLP(3072-1024-1024-10) on MNIST, Fashion MNIST, and Cifar-10. The proposed method can achieve a higher compression rate than existing methods with a lower loss of accuracy. A large number of experiments have proved the reliability and advantages of the proposed method, which can compress the network more effectively, obtain a better lightweight structure, and achieve better performance. The performance improvement in the experimental results also proves that automatic compression of neural networks using DRL is effective. Moreover, using the proximal gradient method to update the DRL can further improve the compression performance.

Author Contributions: Writing—original draft, M.W.; Investigation, J.T.; Formal analysis, H.Z.; Methodology, Z.L.; Supervision, S.X.; Writing—review and editing, M.W. All authors have read and agreed to the published version of the manuscript.

Funding: This research is supported in part by National Natural Science Foundation of China under Grants 62273106, 62203122, in part by China Postdoctoral Science Foundation funded project 2022M720840, and in part by Foreign Expert Project of Ministry of Science and Technology of China under Grants DL2022030011L, G2022030041L.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: All the data and models employed and/or generated during the study appear in the submitted article.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Brock, A.; De, S.; Smith, S.L.; Simonyan, k. High-performance large-scale image recognition without normalization. In Proceedings of the International Conference on Machine Learning, PMLR, Virtual, 18–24 July 2021; pp. 1059–1071.
2. Ding, X.; Xia, C.; Zhang, X.; Chu, X.; Han, J.; Ding, G. Repmlp: Re-parameterizing convolutions into fully-connected layers for image recognition. *arXiv* **2021**, arXiv:2105.0. [CrossRef]
3. Liang, W.; Long, J.; Li, K.C.; Xu, J.; Ma, N.; Lei, X. A fast defogging image recognition algorithm based on bilateral hybrid filtering. *ACM Trans. Multimed. Comput. Commun. Appl. (TOMM)* **2021**, *17*, 1–16. [CrossRef] [CrossRef]
4. Zhang, W.; Wang, J.; Lan, F. Dynamic hand gesture recognition based on short-term sampling neural networks. *IEEE/CAA J. Autom. Sin.* **2021**, *8*, 110–120. [CrossRef] [CrossRef]
5. Kim, S.; Hwang, S.; Hong, S.H. Identifying shoplifting behaviors and inferring behavior intention based on human action detection and sequence analysis. *Adv. Eng. Inf.* **2021**, *50*, 101399. [CrossRef] [CrossRef]
6. Yang, W.; Zhang, T.; Mao, Z.; Zhang, Y.; Tian, Q.; Wu, F. Multi-scale structure-aware network for weakly supervised temporal action detection. *IEEE Trans. Image Process.* **2021**, *30*, 5848–5861. [CrossRef] [CrossRef] [PubMed]
7. Erhan, D.; Szegedy, C.; Toshev, A.; Anguelov, D. Scalable object detection using deep neural networks. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Columbus, OH, USA, 23–28 June 2014; pp. 2147–2154.

8. Cai, Z.; Fan, Q.; Feris, R.S.; Vasconcelos, N. *A Unified Multi-Scale Deep Convolutional Neural Network for Fast Object Detection*; Springer: Berlin/Heidelberg, Germany, 2016; pp. 354–370.
9. Muzahid, A.A.M.; Wan, W.; Sohel, F.; Wu, L.; Hou, L. CurveNet: Curvature-Based Multitask Learning Deep Networks for 3D Object Recognition. *IEEE/CAA J. Autom. Sin.* **2021**, *8*, 1177–1187. [CrossRef] [CrossRef]
10. Long, J.; Shelhamer, E.; Darrell, T. Fully convolutional networks for semantic segmentation. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Boston, MA, USA, 7–12 June 2015; pp. 3431–3440.
11. Arbeláez, P.; Hariharan, B.; Gu, C.; Gupta, S.; Bourdev, L.; Malik, J. Semantic segmentation using regions and parts. In Proceedings of the 2012 IEEE Conference on Computer Vision and Pattern Recognition, Providence, RI, USA, 16–21 June 2012; pp. 3378–3385.
12. Peng, C.; Zhang, X.; Yu, G.; Luo, G.; Sun, J. Large kernel matters–improve semantic segmentation by global convolutional network. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Honolulu, HI, USA, 22–25 July 2017; pp. 4353–4361.
13. Chen, W.; Wilson, J.; Tyree, S.; Weinberger, K.; Chen, Y. Compressing neural networks with the hashing trick. In Proceedings of the International Conference on Machine Learning, PMLR, Lille, France, 6–11 July 2015; pp. 2285–2294.
14. Wang, K.; Liu, Z.; Lin, Y.; Lin, J.; Han, S. Haq: Hardware-aware automated quantization with mixed precision. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Long Beach, CA, USA, 15–20 June 2019; pp. 8612–8620.
15. Kim, H.; Khan, M.U.K.; Kyung, C.M. Efficient neural network compression. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Long Beach, CA, USA, 15–20 June 2019; pp. 12569–12577.
16. Denton, E.L.; Zaremba, W.; Bruna, J.; LeCun, Y.; Fergus, R. Exploiting linear structure within convolutional networks for efficient evaluation. In Proceedings of the Advances in Neural Information Processing Systems, Montreal, QC, Canada, 8–13 December 2014; pp. 1269–1277.
17. Mirzadeh, S.I.; Farajtabar, M.; Li, A.; Levine, N.; Matsukawa, A.; Ghasemzadeh, H. Improved knowledge distillation via teacher assistant. *AAAI Conf. Artif. Intell.* **2020**, *34*, 5191–5198. [CrossRef]
18. Wang, D.; Li, Y.; Wang, L.; Gong, B. Neural networks are more productive teachers than human raters: Active mixup for data-efficient knowledge distillation from a blackbox model. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Seattle, WA, USA, 13–19 June 2020; pp. 1498–1507.
19. Tavakolian, M.; Tavakoli, H.R.; Hadid, A. Awsd: Adaptive weighted spatiotemporal distillation for video representation.In Proceedings of the IEEE/CVF International Conference on Computer Vision, Seoul, Republic of Korea, 27 October–2 November 2019; pp. 8020–8029.
20. Zhao, H.; Wu, J.; Li, Z.; Chen, W.; Zheng, Z. Double Sparse Deep Reinforcement Learning via Multilayer Sparse Coding and Nonconvex Regularized Pruning. *IEEE Trans. Cybern.* **2022**, 1–14 . [CrossRef] [CrossRef] [PubMed]
21. Li, Z.; Su, W.; Xu, M.; Yu, R.; Niyato, D.; Xie, S. Compact Learning Model for Dynamic Off-Chain Routing in Blockchain-Based IoT. *IEEE J. Sel. Areas Commun.* **2022**, *40*, 3615–3630. [CrossRef] [CrossRef]
22. Chen, C.; Tung, F.; Vedula, N.; Mori, G. Constraint-aware deep neural network compression. In Proceedings of the European Conference on Computer Vision (ECCV), Munich, Germany, 8–14 September 2018; pp. 400–415.
23. Tibshirani, R. Regression shrinkage and selection via the lasso. *J. R. Stat. Soc. Ser. B (Methodol.)* **1996**, *58*, 267–288. [CrossRef] [CrossRef]
24. Vettam, S.; John, M. Regularized deep learning with a non-convex penalty. *arXiv* **2019**, arXiv:1909.05142. [CrossRef]
25. Scardapane, S.; Comminiello, D.; Hussain, A.; Uncini, A. Group sparse regularization for deep neural networks. *Neurocomputing* **2017**, *241*, 81–89. [CrossRef] [CrossRef]
26. Yoon, J.; Hwang, S.J. Combined group and exclusive sparsity for deep neural networks. In Proceedings of the International Conference on Machine Learning, PMLR, Sydney, Australia, 6–11 August 2017; pp. 3958–3966.
27. Louizos, C.; Welling, M.; Kingma, D.P. Learning sparse neural networks through L_0 regularization. *arXiv* **2017**, arXiv:1712.01312. [CrossRef]
28. Xiao, X.; Wang, Z. Autoprune: Automatic network pruning by regularizing auxiliary parameters. In Proceedings of the Advances in Neural Information Processing Systems 32 (NeurIPS 2019), Vancouver, BC, Canada, 8–14 September 2019; Volume 32.
29. Liu, Z.; Sun, M.; Zhou, T.; Huang, G.; Darrell, T. Rethinking the value of network pruning. *arXiv* **2018**, arXiv:1810.05270. [CrossRef]
30. He, Y.; Lin, J.; Liu, Z.; Wang, H.; Li, L.J.; Han, S. Amc: Automl for model compression and acceleration on mobile devices. In Proceedings of the European Conference on Computer Vision (ECCV), Munich, Germany, 8–14 September 2018; pp. 784–800.
31. Ng, A.Y.; Harada, D.; Russell, S. Policy invariance under reward transformations: Theory and application to reward shaping. *Icml* **1999**, *99*, 278–287.
32. Mnih, V.; Kavukcuoglu, K.; Silver, D.; Graves, A.; Antonoglou, I.; Wierstra, D.; Riedmiller, M. Playing atari with deep reinforcement learning. *arXiv* **2013**, arXiv:1312.5602. [CrossRef]
33. Gupta, M.; Aravindan, S.; Kalisz, A.; Chandrasekhar, V.; Jie, L. Learning to Prune Deep Neural Networks via Reinforcement Learning. *arXiv* **2020**, arXiv:2007.04756. [CrossRef]
34. Watkins, C.J.; Dayan, P. Q-learning. *Mach. Learn.* **1992**, *8*, 279–292. [CrossRef] [CrossRef]
35. Hasselt, H. Double Q-learning. *Adv. Neural Inf. Process. Syst.* **2010**, *23*, 2613–2621.
36. Lillicrap, T.P.; Hunt, J.J.; Pritzel, A.; Heess, N.; Erez, T.; Tassa, Y.; Silver, D.; Wierstra, D. Continuous control with deep reinforcement learning. *arXiv* **2015**, arXiv:1509.02971. [CrossRef]

37. Schulman, J.; Wolski, F.; Dhariwal, P.; Radford, A.; Klimov, O. Proximal policy optimization algorithms. *arXiv* **2017**, arXiv:1707.06347. [CrossRef]
38. Pizzocaro, F.; Torreggiani, D.; Gilardi, G. Inhibition of apple polyphenoloxidase (PPO) by ascorbic acid, citric acid and sodium chloride. *J. Food Process. Preserv.* **1993**, *17*, 21–30. [CrossRef] [CrossRef]
39. Qiu, S.; Yang, Z.; Ye, J.; Wang, Z. On finite-time convergence of actor-critic algorithm. IEEE *J. Sel. Areas Inf. Theory* **2021**, *2*, 652–664. [CrossRef] [CrossRef]
40. Wen, J.; Kumar, S.; Gummadi, R.; Schuurmans, D. Characterizing the gap between actor-critic and policy gradient. In Proceedings of the International Conference on Machine Learning, PMLR, Virtual, 18–24 July 2021; pp. 11101–11111.
41. Li, L.; Li, D.; Song, T.; Xu, X. Actor–Critic Learning Control With Regularization and Feature Selection in Policy Gradient Estimation. *IEEE Trans. Neural Networks Learn. Syst.* **2020**, *32*, 1217–1227. [CrossRef] [CrossRef]