Neural Network Panning: Screening the Optimal Sparse Network Before Training

Xiatao Kang[1], Ping Li[1,3,*], Jiayi Yao[1], and Chengxi Li[2]

[1] School of Computer and Communication Engineering, Changsha University of Science and Technology
[2] School of Computer, Xidian University
[3] Hunan Provincial Key Laboratory of Intelligent Processing of Big Data on Transp
kangxiatao@gmail.com, lping9188@163.com

Abstract. Pruning on neural networks before training not only compresses the original models, but also accelerates the network training phase, which has substantial application value. The current work focuses on fine-grained pruning, which uses metrics to calculate weight scores for weight screening, and extends from the initial single-order pruning to iterative pruning. Through these works, we argue that network pruning can be summarized as an expressive force transfer process of weights, where the reserved weights will take on the expressive force from the removed ones for the purpose of maintaining the performance of original networks. In order to achieve optimal expressive force scheduling, we propose a pruning scheme before training called Neural Network Panning which guides expressive force transfer through multi-index and multi-process steps, and designs a kind of panning agent based on reinforcement learning to automate processes. Experimental results show that Panning performs better than various available pruning before training methods. Our code is made public at: https://github.com/kangxiatao/RLPanning.

Keywords: Deep Learning · Reinforcement Learning · Network Pruning · Pruning Before Training.

1 Introduction

In recent years, neural networks have achieved breakthrough results in computer vision[15,40] and natural language processing[16]. More complex architectural designs improve model performance but significantly increase the number of parameters of the neural network. Research on neural network compression[10] is devoted to reducing the memory overhead and computational cost of neural networks and maintaining the performance of the network architecture. Neural network pruning[11] is the most direct method of neural network compression, and the purpose of compression is achieved by filtering out unimportant weights.

The latest research focuses on network pruning before training[17,12,5], with progress in theory and application. A series of studies of the lottery ticket hypothesis[4,6,25] demonstrates the effectiveness of pruning before training. Many

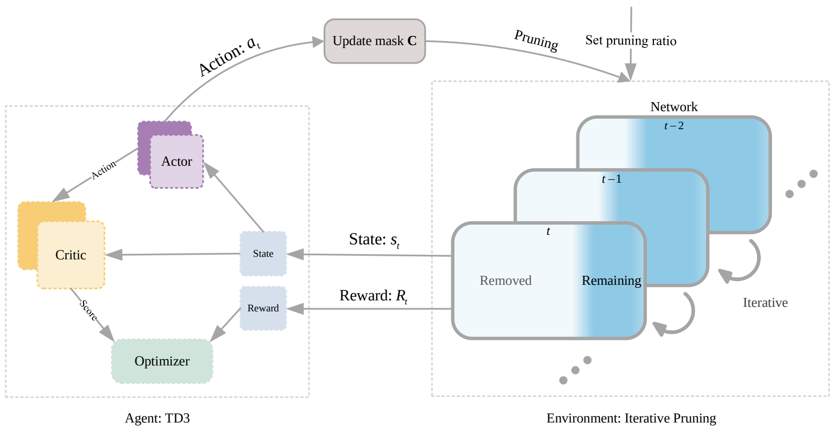

Fig. 1: Overview of Panning based on reinforcement learning. On the right is the Panning environment, which iteratively prunes the network after setting the pruning rate, and the feedback includes the current state of the sparse network and the set reward. On the left is the TD3 agent, which acts through the feedback given by the environment, that is, it selects the optimal pruning strategy based on the network state during iterative pruning.

methods of pruning before training achieve desirable performance[14,33,34]. However, some work argues that pruning before training should be considered a search of the network architecture[24] and that random selection of weights with equal layer pruning quotas[5,32] can even achieve better generalization. Drop weights randomly or by structure from left to right during training using sparse constraints[1], and both are very effective. We believe that the idea of these works is that weights are equal, and the network structure is directly examined when pruning the network rather than the weight ordering under a particular index.

The characterization of weight expressiveness has always been the research focus of neural network pruning algorithms. Using sensitivity[18] as a metric is currently the mainstream method for pruning before training. In our research, we found that the process of pruning can be regarded as the process of transferring the expressive force of the removed weight to the remaining weight so that any weight has an equal opportunity to express. Assuming that the weight sensitivity is a representation of the weight expressive force, Figure 2 shows the expressive force transfer through iterative pruning. Different measurement methods indicate different affordability of weights, and a single measurement index is always flawed. Loss-sensitive correlation metrics[18,35] are prone to inter-layer disconnection at high compression rates, while metrics that only consider inter-layer equilibrium[33] do not generalize well at low compression rates. Some methods improve performance using iterative pruning with a single metric and adding constraints[2,34], but this benefits from the fact that iterative pruning examines structural changes and does not improve the one-sidedness of the single metric.

Therefore, we propose a multi-metric, multi-process progressive pruning method based on our improvements and existing metrics. The process of picking weights is like a gold purification process, which we call Panning. Since dynamic iterative pruning will generate many state change trajectories, we design Panning as an environment with actions and feedback, and use reinforcement learning to search for Panning steps(called RLPanning), that is, the agent automates the Panning process through policy learning(Figure 1). Unlike simple metric ranking, RLPanning examines multiple expressive forces during pruning and determines whether and when the forces can be transferred according to the state of the target network in terms of loss variation, sparsity, and interlayer structure to maximize the benefits and obtain the ideal sparse network.

The contributions of this paper are as follows:

- We generalize the pruning process as the expressive force transfer process, analyze various weight metrics, and make improvements.
- We propose a multi-metric, multi-process iterative pruning method before training, called Panning. Panning performs well by pruning the network before training, maintaining generalization well under extremely high compression rates.
- We design a Panning environment and use reinforcement learning to learn a Panning strategy by sampling spatial actions, called RLPanning. RLPanning obtains the optimal pruning strategy with a comprehensive measure of multiple indicators through dynamic expressive force transfer.

2 Related work

Neural network pruning has been a hot research topic along with the development of artificial intelligence so far. The purpose of pruning is to reduce the parameters and computation of neural networks to achieve compression of neural networks[10], which can be divided into two major categories of unstructured and structured pruning according to the way of pruning networks. Unstructured pruning performs fine-grained pruning intending to minimize the number of parameters, and the classical process is training, pruning, and fine-tuning[11]. Sparse constraints in training to obtain sparse structures are also more widely used[37,27]. Structured pruning focuses on the structure of the network, and the pruning results in a lean, compact model. In convolutional networks, structured pruning is usually considered the pruning of the filter[19], and most of the work sets structured constraints[8,23,36] in training to prune the network.

In order to obtain better compression structure and performance, many works are devoted to automatic neural network architecture search[31,39,38]. Although automatic search network architecture is prevalent, it also suffers from substantial computational overhead due to the infinite search space. Subsequent work has made new attempts to use adversarial learning[21] or reinforcement learning[13] to search for pruning strategies under a fixed framework to achieve network simplification.

Pruning before training. Pruning before training has substantial theoretical and applied value. The lottery ticket hypothesis[4] demonstrates the possibility of obtaining superior sparse networks before training through extensive experiments. Since the lottery hypothesis was put forward, much work has been done on expansion[3,6,28] and theoretical research[25,29].

In the current application, the pruning before training focuses on fine-grain pruning, and the weight is usually measured by examining the effect of removing a certain weight on a certain state of the network. Specifically, SNIP[18] examines the impact of pruning weights on loss, that is, the sensitivity of weights to model losses. GraSP[35] examines the degree to which the removal weight decreases the loss, that is, the sensitivity of the weight to the change in loss. SynFlow[33] starts from the trainability of the maximum compression network and examines the balance of weights between layers. FORCE[14] progressively prunes the network based on the SNIP method and allows for the resurrection of removed weights, taking into account changes in the network structure. Among them, SNIP and GraSP methods can obtain sparse networks with only a single network pruning. Methods that consider inter-layer balance and network structure require iterative network pruning. In addition, there are iterative pruning methods to make dynamic expansion[2,22] or set constraints[34] to obtain good performance.

For weight measurement, SynFlow summarizes the mathematical expression of sensitivity, called synaptic saliency[33]$\mathcal{S}(\theta) = \frac{\partial \mathcal{R}}{\partial \theta} \odot \theta$, where $\mathcal{R}$ is a scalar loss function, and θ is a parameter to be examined. Hence $\mathcal{S}_{SNIP}(\theta) = \left| \frac{\partial \mathcal{L}}{\partial \theta} \odot \theta \right|, \mathcal{S}_{GraSP}(\theta) = \frac{\partial g^T g}{\partial \theta} \odot \theta$, $g^T g$ is the first-order Taylor approximation of loss drop.

3 Methodology

In this section, we introduce Panning in detail and then automate Panning using reinforcement learning. Our goal is to have the agent schedule the expressive force of the weights according to the state of the network, cross-validate the weights with a variety of meaningful metrics, and find the optimal sparse sub-network before training. Finally, we discuss and improve the existing metrics and apply them in Panning.

3.1 Problem definition

We mainly consider fine-grained pruning of neural networks, that is, removing parts of weights that are considered unimportant, regardless of the structure of the network. In order to describe this irregular sparse network conveniently, it is usually represented by the Hadamard product of the mask $\mathbf{c}$ and the weight $\mathbf{w}$, then the optimization problem of the network is:

$$\begin{gathered}\min_{\mathbf{c},\mathbf{w}} \mathcal{L}(\mathbf{c} \odot \mathbf{w}; \mathcal{D}) \\ \text{s.t. } \mathbf{c} \in \{0,1\}^m, \|\mathbf{c}\|_0 \leq m(1-\rho)\end{gathered} \tag{1}$$

Where $\mathcal{L}(\cdot)$ represents the loss function, $\mathcal{D}$ is the dataset, m is the total number of weights in the network, and ρ is the target pruning ratio (sparse degree). When the mask $\mathbf{c}$ corresponding to the weight is set to zero, it means that the weight is removed.

3.2 Transfer of expressive force

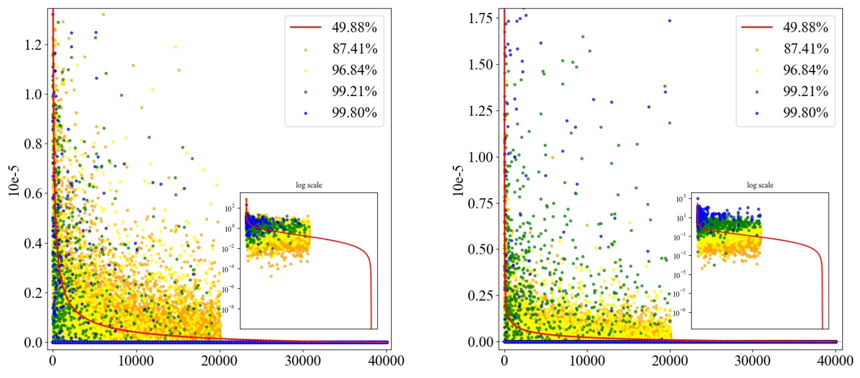

Fig. 2: Comparison of metric scores at different pruning ratios during the iterative pruning process of SNIP (left) and GraSP (right). Experiments were performed under VGG19/CIFAR10. The vertical axis is the metric score $\mathcal{S}$ (where $\mathcal{S}_{GraSP}$ is the absolute value), and the horizontal axis is the number of weights (in order to reduce the amount of data, after flattening all weights into vectors, take a value every 500 weights). After sorting the weight scores for the first time (line), the score changes (dots) under other pruning ratios are displayed at the corresponding weight positions. The subplots are the logarithmic changes in the vertical axis scale.

We use SNIP and GraSP for iterative pruning, respectively, and show in Figure 2 the changes in the metric scores of all weights in the network under different compression ratios (the two adjacent pruning rates can represent the change in scores before and after pruning), and more obvious hierarchical changes can be observed after taking the logarithm of the vertical axis (as shown in the subplot). We find that the weight expressive force is sensitive to the compression ratio. As the compression ratio increases, some of the less expressive weight metrics improve significantly (above the red line), and vice versa.

Iterative dynamic pruning is widely considered to perform better, especially before training. Through continuous compression and gradient iterations, weights that were evaluated as strong in performance in the most recent iteration are retained. The invariant properties of flow transported between layers are given in the SynFLow[33]. Naturally, we view model compression as a process of transferring expressive force between different weights based on flow sensitivity. Take $\mathcal{S}(\theta) = |\frac{\partial \mathcal{R}}{\partial \theta} \odot \theta|$ as the weight expressive force, and only observe the magnitude of the force.

Specifically, the expressive force of the removed weight will be transferred to the remaining weight, and the force of the remaining part will also be adjusted due to structural changes. Therefore, weights with low importance at the beginning may become critical weights in the future, and a single pruning can easily remove this part of the weights, resulting in the unreasonable distribution of weight expressive force. It is challenging to maintain layer balance even if layer collapse does not occur. The advantage of iterative pruning is that it gradually transfers the expressive force and examines the changes in the structure so that the connection structure between the remaining weights is better maintained.

3.3 Artificial Panning

Although iterative pruning solves the layer collapse problem, existing iterations use a single metric pruning throughout, ignoring the expressive force of weights under other metrics. Simply summing up multiple metrics for measurement will easily cover up specific properties, resulting in an indistinct degree of weight discrimination. We design the Panning algorithm based on dynamic iterations to establish multiple mappings between sparse network structure changes and weights. The characteristic of Panning is to observe the state of the network in the iterative process, and select the most favorable indicator to measure the weight according to the state change. Based on this, the network is progressively pruned in a loop, and the removed weights are allowed to re-live until the desired sparse network is finally obtained.

Specifically, the pruning rate changes exponentially during the iteration process, and the pruning ratio ρ_i of round i is:

$$\rho_i = 1 - (1 - \rho_{target})^{\left(\frac{i}{T}\right)} \tag{2}$$

where ρ_{target} is the target pruning ratio, and T is the total the number of iterations. Then calculate the various metrics in the current state of the model, and obtain the score of the weight $\mathbf{w}$ through a specific fusion:

$$\mathcal{S}_{\mathbf{w}} = \sum_{i=1}^{k} p_i \mathcal{N}\left(\mathcal{S}_i\right) \tag{3}$$

Algorithm 1 Panning

```
Require: Weight $\mathbf{w}$, batch data $\mathcal{D}^b$, target pruning rate $\rho_{target}$.
 1: Initialize $\mathbf{w}$, $\mathbf{c}$
 2: for $i = 0$ to $T$ do
 3:     Remove weight $\mathbf{w}^* = \mathbf{c} \odot \mathbf{w}$
 4:     Calculate loss $\mathcal{R}$
 5:     Reset weight $\mathbf{w}^* = \mathbf{w}$
 6:     Calculate score $\mathcal{S}_{\mathbf{w}} = \sum_{i=1}^{k} p_i \mathcal{N}(\mathcal{S}_i)$
 7:     Update pruning rate $\rho_i = 1 - (1 - \rho_{target})^{\left(\frac{i}{T}\right)}$
 8:     Update mask $\mathbf{c}$
 9: end for
10: return $\mathbf{c}$
```

where p_i is a hyperparameter that adjusts the proportion of the metric, and $\mathcal{N}(\mathcal{S}_i)$ represents the normalization of the metric score. Finally, the network is gradually pruned according to the score $\mathcal{S}_{\mathbf{w}}$ and the pruning ratio ρ_i:

$$\mathbf{c} = \mathcal{S}_{\mathbf{w}}(\boldsymbol{\theta}) > \mathcal{S}_{top(m(1-\rho_i))} \tag{4}$$

where $\mathcal{S}_{top(\kappa)}$ is the κ-th value after the descending order, and m is the total number of weights. When the weight $\mathbf{w}_i$ does not satisfy the conditions of the above formula, the corresponding mask $\mathbf{c}_i$ is set to zero, the weight is removed and exists as a zero value. The value of $\mathbf{w}_i$ is reset the next time the weight sensitivity $\mathcal{S}$ is calculated, so that the removed weight can be restored. Algorithm 1 shows the complete gold panning steps.

3.4 Panning based on reinforcement learning

According to our observation, an ideal sparse network can be obtained by setting the hyperparameters of Panning based only on the compression of the network. Naturally, if more network states are examined and dynamic expressive force transfer is achieved based on state changes, this can theoretically lead to a sparse network with better performance. Therefore, we design Panning as an environment containing action and state space, so that the intelligent body can automate Panning through policy learning. Figure 1 gives an overview of RLPanning, which is described in detail in this section.

Panning environment. Each Panning iteration results in a new sparse network, that is, different sparsity and performance. The number of iterations is taken as the time t, so the state space s_t of the Panning environment is:

$$s_t = (\mathcal{L}, \varDelta\mathcal{L}, \mathcal{L}_s, \varDelta\mathcal{L}_s, \rho_t, \rho_e, t) \tag{5}$$

where $\mathcal{L}_s, \varDelta\mathcal{L}_s$ are the loss and loss reduction of the sparse sub-network in the current state, respectively, ρ_e represents the effective compression rate of the network[34] (Invalid retention due to disconnection between layers is prone to occur at high compression rates). Note that all states are normalized to facilitate agent training. The action space a of Panning is the hyperparameter p_i that regulates the proportion of metrics, and assuming that there are κ metrics, the action space dimension is κ and is continuous for the action:

$$a \in [-1, 1], p_i = \frac{a_i + 1}{2} \tag{6}$$

Our target is to affect $\mathcal{L}, \Delta\mathcal{L}$ as little as possible during pruning and to ensure that the effective compression rate is close to the target compression rate. So we design a reward term R_t consisting of four components:

$$\begin{aligned} R_t &= -\left|\mathcal{N}(\mathcal{L}) - \mathcal{N}\left(\mathcal{L}_s\right)\right| - \left|\mathcal{N}(\Delta\mathcal{L}) - \mathcal{N}\left(\Delta\mathcal{L}_s\right)\right| - \alpha\left|\rho_e - \rho_t\right| - r_{\text{done}} \\ r_{\text{done}} &= \begin{cases} T - t, & \text{if } \rho_e = 1 \\ 0, & \text{otherwise} \end{cases} \end{aligned} \tag{7}$$

where $\mathcal{N}(\cdot)$ represents normalization. T is the maximum number of iterations. r_{done} is the reward in the final state, $done = True$ when $t = T$ or effective compression ratio $\rho_e = 1$, $\rho_e = 1$ means that all weights are removed, and the iteration will end early. α is the reward strength that adjusts the quality of network compression. Usually the ineffective compression weight is less than one percentage point, and we set α to 100.

Panning agent. The continuous action of Panning is a deterministic strategy, and we adopt TD3[7], which performs well in the field of continuous control, as the Panning agent. TD3 is an actor-critic network architecture. As an optimized version of DDPG[20], the delay and smoothing processing of TD3 largely mitigates the effects of sudden changes in environmental states due to different batches of samples, which is well suited to our Panning environment.

The workflow of the TD3 agent is shown on the left in Figure 1. The actor network gives an action, and the critic network estimates the reward for the action. The critic network is equivalent to the Q network in DQN[26], and the goal is to solve the action a that maximizes the Q value. The actions and states of the Panning environment are all standardized, the structure and optimization of the TD3 agent basically do not need to be changed, and the network scale and all hyperparameters are shown in the experimental part.

In addition, after resetting the environment, in order to better sample in the space, the final target compression ratio ρ_T is randomly selected between $\{80\%, 99.99\%\}$. In addition, we have set a curriculum learning[30] from easy to complex, that is, the probability of the compression target ρ_T being a low compression ratio in the early stage of the agent training is high, and the probability of $\rho_T > 99\%$ in the later stage is higher, thereby speeding up the convergence speed of the agent training.

3.5 Selection of metrics

The currently widely accepted singe-shot pruning metrics are SNIP and GraSP, which are very sample-dependent and perform poorly at high compression ratios. SynFlow is more prominent in iterative pruning and achieves a good balance between layers. The most significant advantage of SynFlow is that it does not require samples, but the lack of consideration of sample characteristics also makes it generalize poorly at low compression rates. Actually, SNIP, GraSP, and SynFlow are complementary and computationally very similar. Therefore, we choose these three metrics to apply to Panning, with action space $\kappa = 3$ and weight score $\mathcal{S}_w = p_1\mathcal{N}\left(\mathcal{S}_{SynFlow}\right) + p_2\mathcal{N}\left(\mathcal{S}_{SNIP}\right) + p_3\mathcal{N}\left(\mathcal{S}_{GraSP}\right)$.

Due to gradient decay in the deep network, when the SNIP pruning ratio is large, it is easy to remove too many subsequent convolutional layers and cause faults. The idea of GraSP is to maintain the gradient flow and prune the weights to maximize the loss drop, which improves compressibility. However, there is a significant error in the calculation of the first-order approximation $g^T g$, especially in the ill-conditioned problem[9] that the first-order term of the loss function is smaller than the second-order term during training. Based on this, we believe that maximizing the gradient norm is not equivalent to maintaining the gradient flow, and it is more accurate to remove weights that have little influence on the gradient flow. Therefore, we changed $\mathcal{S}_{GraSP}$ to $|\frac{\partial g^T g}{\partial \theta} \odot \theta|$, that is, to examine the impact of weights on model trainability. If it is analyzed from the perspective of acting force, after taking the absolute value, it means that only the magnitude of the force is considered, and the direction of action of the force is ignored. Moreover, the experiment proves that the modified $\mathcal{S}_{GraSP}$ has better performance.

In addition, in the classification task, when $\mathcal{R}$ is the fitting loss $\mathcal{L}$ of the network, the gradient information brought by different samples is more abundant. Therefore, when calculating the loss $\mathcal{L}$, we set a specific sampling to ensure that $\mathcal{D}^b$ contains all categories and an equal number of samples for each category. Suppose the dataset has l categories, k samples are taken from each category, then $\mathcal{D}^b = \left\{\left(x^b, y^b\right)\right\},\ x^b = \left(x^1_{1,2,\ldots,k}, x^2_{1,2,\ldots,k}, \cdots, x^l_{1,2,\ldots,k}\right)$.

4 Experiments

4.1 Experimental setup

In the image classification task, we prune LeNet5, VGG19, and ResNet18 convolutional networks with panning before training and select MNISIT, Fashion-MNISIT, CIFAR10/100, and TinyImageNet datasets to evaluate our method. In this experiment, images are enhanced by random flipping and cropping. Details and hyperparameters of sparse network training after pruning are shown in Table 1.

4.2 Performance of the improved metric

The performance comparison between the improved SNIP and GraSP metrics and the original method on VGG19/CIFAR10 is shown in Table 2. The results were obtained by taking the average of three experiments.

Table 1: Model optimizer and hyperparameters.

	MNIST	CIFAR	ImageNet	
	LeNet	VGG/ResNet	VGG	ResNet
Optimizer	Momentum (0.9)			
Learning Rate	Cosine Annealing (0.1)			
Training Epochs	80	180	200	300
Batch Size	256	128	128	128
Weight Decay	1e-4	5e-4	5e-4	1e-4

Table 2: Performance comparison of improved SNIP and GraSP.

VGG19/CIFAR10	Acc: 94.20%				
Pruning ratio	85%	90%	95%	98%	99%
SNIP	93.91	93.82	93.72	91.16	10.00
Ours SNIP	**94.05**	**93.98**	**93.86**	**91.83**	10.00
GraSP	93.59	93.50	92.90	92.39	91.04
Ours GraSP	**93.77**	**93.69**	**93.48**	**92.78**	**91.84**

Note that the loss is computed using a batch of samples containing all labels, although the overall improved SNIP has only a slight improvement in accuracy, which is essential on datasets with more complex labels. Under the 99% pruning rate, the test set accuracy is still 10%, which is caused by the fault of SNIP itself. For the improvement of GraSP, the improvement of accuracy is more prominent, especially after the compression rate is higher than 90%. This suggests that sensitivity to the gradient norm is better than maximizing the gradient norm, validating our previous analysis.

4.3 Experimental results of artificial Panning

In Section 3.5, we introduce the metrics selected by Panning. In the process of Panning, when the compression rate is low, the balance between layers is better, and the weight sensitive to loss should have the highest priority. When the compression rate is high, the number of weights is minimal, and it is easy to have connectivity matters[34], so the sensitivity between layers is more important. Combined with the influence of gradient norm on trainability, the hyperparameter settings under different compression stages in the Panning process are shown in Table 3.

We compare the compression ratio of Panning, SynFlow, SNIP iteration, and GraSP iteration between $\{10^{-1}, 10^{-4}\}$. In order to highlight the effectiveness of Panning, the SNIP and GraSP iterations also use the FORCE[14] method to prune the network dynamically, and the number T of iterative pruning is unified

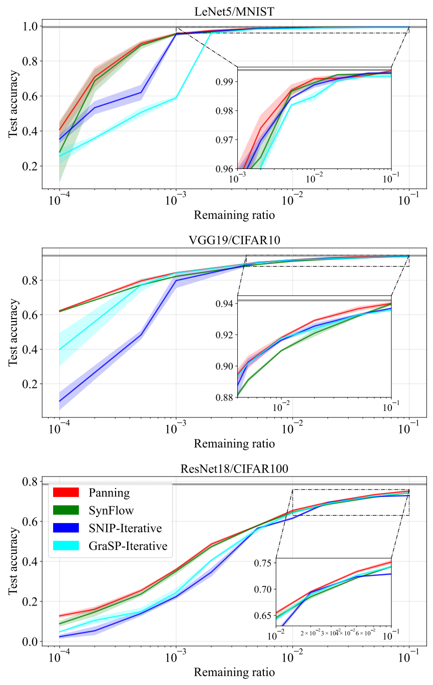

Fig. 3: Performance comparison of Panning and iterative pruning. The shaded area indicates the error of repeated experiments. The gray horizontal line is the baseline.

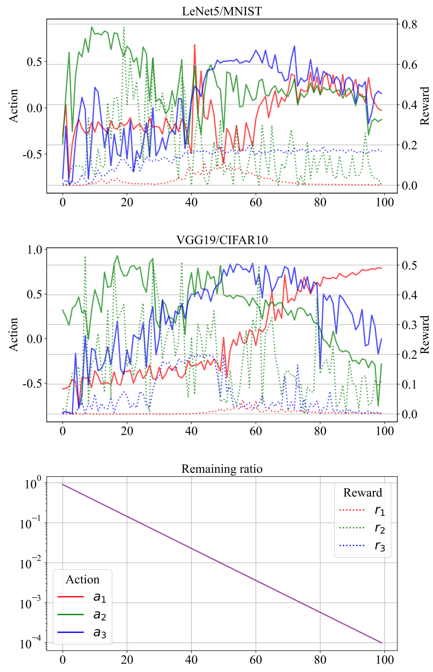

Fig. 4: Action and reward changes generated by RLPanning during pruning. The left axis is the output action a of the agent. The right axis is the reward r of environmental feedback. The bottom graph shows the change in the remaining ratio over iterations.

as 100 times. Taking the L2 regularized complete network as the baseline, Figure 3 shows the experimental results on VGG19/CIFAR10, ResNet18/CIFAR100, and LeNet5/MNIST.

Note that the curve corresponding to Panning in the figure is always above the other methods, whether with a high or low compression ratio. The MNIST dataset is relatively simple, and the differences between the various methods are minor. The advantage of Panning on VGG19/CIFAR10 is mainly manifested in the low compression rate. On ResNet18/CIFAR100, the generalization ability through Panning is also maintained well at extremely high compression rates. In addition, the hyperparameters that adjust the proportion of metrics in Panning are set based on our estimates. In fact, if the parameters in Table 3 are further tuned, Panning will achieve better performance, but our estimated settings have demonstrated the effectiveness of Panning for multiple screening.

Table 3: Hyperparameter setting of artificial Panning.

Pruning ratio	p_1(SynFlow)	p_2(SNIP)	p_3(GraSP)
$(0, 0.8]$	0.2	0.5	0.3
$(0.8, 0.9]$	0.2	0.4	0.4
$(0.9, 0.98]$	0.2	0.3	0.5
$(0.98, 0.99]$	0.4	0.2	0.4
$(0.99, 1)$	0.5	0	0.5

Table 4: TD3 training details and hyperparameters.

Training		TD3	
Optimizer	Adam	Exploration Noise	0.1
Learning Rate	3e-4	Discount Factor	0.99
Start Timesteps	2e3	Network Update Rate	0.01
Max Timesteps	2e5	Policy Noise	0.2
Batch Size	256		

4.4 Experimental results of RLPanning

The actor and critic networks of the TD3 agent in RLPanning are both three-layer linear networks, using the Tanh activation function, and the dimension of the hidden layer is 256. The training details and key hyperparameters are shown in Table 4. When training the agent, we only use LeNet5/MNIST as the network and dataset for the Panning environment and do not need other network structures and datasets. The resulting Panning agent can be adapted to other convolutional network pruning tasks.

We apply the trained Panning agent to prune LeNet5 and VGG19. The feedback of the Panning environment and the actions of the Panning agent are shown in Figure 4. Where $\{r_1, r_2, r_3\}$ correspond to the first three unsigned items in Equation (7), respectively. The reward r comes from the state of the sparse network, and the r_2 change is more drastic than other reward items due to different data batches. It is not difficult to find from the figure that the action a_2 plays a leading role in the early stage of the iteration. As the remaining weights decrease, the magnitudes of actions a_1, a_3 begin to increase. By the late iteration, the reward r gradually stabilizes, with comparable action magnitude in LeNet5/MNIST and even much higher action a_1 in VGG19/CIFAR10, due to the deeper network layers of VGG and the more significant challenge of inter-layer equalization. Note that the actions produced by the agent are roughly consistent with our preset changes, and the same pattern occurs on untrained networks and datasets.

The performance of RLPanning pruned LeNet5 and VGG19 is shown in Table 5. We take the average of three experiments to get the results. It is observed

Table 5: Performance comparison of RLPanning and Panning.

LeNet5/MNIST	Acc: 99.40%					
Pruning ratio	90%	95%	98%	99%	99.9%	99.99%
Panning	99.37	**99.21**	99.11	99.09	95.51	40.85
RLPanning	**99.40**	99.18	**99.23**	**99.15**	**97.89**	**45.43**
LeNet5/FashionMNIST	Acc: 91.98%					
Panning	90.67	90.14	88.74	86.92	68.97	26.67
RLPanning	**90.84**	**90.53**	**89.35**	**87.41**	**69.02**	**30.26**
VGG19/CIFAR10	Acc: 94.20%					
Panning	94.02	93.66	92.81	91.75	83.87	62.32
RLPanning	**94.09**	**93.80**	**92.98**	**92.12**	**84.52**	**64.51**

Table 6: Performance comparison of pruned VGG19 and ResNet18 on Tiny-ImageNet.

Network	VGG19: 63.29%			ResNet18: 63.92%		
Pruning ratio	90%	95%	98%	90%	95%	98%
SNIP	61.15	59.32	49.04	60.42	58.56	50.66
GraSP	60.26	59.53	56.54	60.18	58.84	55.79
SynFlow	59.54	58.06	45.29	59.03	56.77	46.34
Panning	61.67	**60.25**	**58.33**	60.74	**58.92**	**56.48**
RLPanning	**61.84**	59.83	58.17	**61.07**	58.73	55.82

that there is less room for improvement at low compression ratios, and RLPanning is very close to Panning in terms of test accuracy. However, RLPanning still shows the advantage of dynamic pruning according to the environment at a high compression rate. Table 6 shows the performance comparison on TinyImageNet. It is observed that Panning has very good performance on large datasets, or rather more obvious improvement in the underfitting state. Note that RLPanning is analogous to Panning on TinyImageNet and significantly outperforms other methods. RLPanning is a little more prominent at low compression ratios. Note that the TD3 model we trained based on LeNet5/MNIST, and it is impossible to accurately capture features for large datasets. But even so, the experimental data verifies the effectiveness of the agent trained on the small model can still be applied to the large model and large dataset. This means that agents trained with reinforcement learning can suppress the interference caused by changes in dataset size and network scale, and can more profoundly reflect the inherent nature of weight screening.

5 Conclusion

In the experiments of this paper, we generalize the pruning process as the process of expressive force transfer, and analyze and improve the existing weight metrics before training. We then propose a pruning before training approach for Panning and automate the Panning process through reinforcement learning. Our experimental results show that Panning further improves the performance of pruned neural networks before training. The expressive force transfer is a fascinating phenomenon, and in the follow-up work, we will consider the higher-order terms of the Taylor approximation to design a new metric. In addition, we will make further explorations in structured sparsity and constraint-guided sparsity based on expressive force transfer.

Acknowledgements This research was partially supported by Hunan Provincial Key Laboratory of Intelligent Processing of Big Data on Transp. Thanks to Google Cloud and Huawei Cloud for cloud computing services.

References

1. Bu, J., Daw, A., Maruf, M., Karpatne, A.: Learning compact representations of neural networks using discriminative masking (DAM). In: NeurIPS. pp. 3491–3503 (2021)
2. Cho, M., Joshi, A., Hegde, C.: ESPN: extremely sparse pruned networks. CoRR **abs/2006.15741** (2020), https://arxiv.org/abs/2006.15741
3. Desai, S., Zhan, H., Aly, A.: Evaluating lottery tickets under distributional shifts. EMNLP-IJCNLP 2019 p. 153 (2019). https://doi.org/10.18653/v1/D19-6117
4. Frankle, J., Carbin, M.: The lottery ticket hypothesis: Finding sparse, trainable neural networks. In: ICLR (2019), https://openreview.net/forum?id=rJl-b3RcF7
5. Frankle, J., Dziugaite, G.K., Roy, D., Carbin, M.: Pruning neural networks at initialization: Why are we missing the mark? In: ICLR (2021), https://openreview.net/forum?id=Ig-VyQc-MLK
6. Frankle, J., Dziugaite, G.K., Roy, D.M., Carbin, M.: Stabilizing the lottery ticket hypothesis. arXiv preprint arXiv:1903.01611 (2019), https://doi.org/10.48550/arXiv.1903.01611
7. Fujimoto, S., van Hoof, H., Meger, D.: Addressing function approximation error in actor-critic methods. In: ICML. Proceedings of Machine Learning Research, vol. 80, pp. 1582–1591. PMLR (2018)
8. Gao, S., Liu, X., Chien, L., Zhang, W., Alvarez, J.M.: VACL: variance-aware cross-layer regularization for pruning deep residual networks. In: ICCV Workshops. pp. 2980–2988. IEEE (2019), https://doi.org/10.1109/ICCVW.2019.00360
9. Goodfellow, I., Bengio, Y., Courville, A.: Deep learning. MIT press (2016)
10. Han, S., Mao, H., Dally, W.J.: Deep compression: Compressing deep neural network with pruning, trained quantization and huffman coding. In: ICLR (2016)
11. Han, S., Pool, J., Tran, J., Dally, W.J.: Learning both weights and connections for efficient neural network. In: NIPS. pp. 1135–1143 (2015)
12. Hayou, S., Ton, J., Doucet, A., Teh, Y.W.: Robust pruning at initialization. In: ICLR (2021), https://openreview.net/forum?id=vXj_ucZQ4hA

13. He, Y., Lin, J., Liu, Z., Wang, H., Li, L., Han, S.: AMC: automl for model compression and acceleration on mobile devices. In: ECCV (7). Lecture Notes in Computer Science, vol. 11211, pp. 815–832. Springer (2018)
14. de Jorge, P., Sanyal, A., Behl, H.S., Torr, P.H.S., Rogez, G., Dokania, P.K.: Progressive skeletonization: Trimming more fat from a network at initialization. In: ICLR (2021), https://openreview.net/forum?id=9GsFOUyUPi
15. Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. In: NIPS. pp. 1106–1114 (2012)
16. Lan, Z., Chen, M., Goodman, S., Gimpel, K., Sharma, P., Soricut, R.: ALBERT: A lite BERT for self-supervised learning of language representations. In: ICLR. OpenReview.net (2020)
17. Lee, N., Ajanthan, T., Gould, S., Torr, P.H.S.: A signal propagation perspective for pruning neural networks at initialization. In: ICLR (2020), https://openreview.net/forum?id=HJeTo2VFwH
18. Lee, N., Ajanthan, T., Torr, P.H.S.: Snip: single-shot network pruning based on connection sensitivity. In: ICLR (Poster) (2019), https://openreview.net/forum?id=B1VZqjAcYX
19. Li, H., Kadav, A., Durdanovic, I., Samet, H., Graf, H.P.: Pruning filters for efficient convnets. In: ICLR (Poster). OpenReview.net (2017)
20. Lillicrap, T.P., Hunt, J.J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D., Wierstra, D.: Continuous control with deep reinforcement learning. In: ICLR (Poster) (2016)
21. Lin, S., Ji, R., Yan, C., Zhang, B., Cao, L., Ye, Q., Huang, F., Doermann, D.S.: Towards optimal structured CNN pruning via generative adversarial learning. In: CVPR. pp. 2790–2799. Computer Vision Foundation / IEEE (2019)
22. Liu, S., Yin, L., Mocanu, D.C., Pechenizkiy, M.: Do we actually need dense overparameterization? in-time over-parameterization in sparse training. In: ICML. Proceedings of Machine Learning Research, vol. 139, pp. 6989–7000. PMLR (2021)
23. Liu, Z., Li, J., Shen, Z., Huang, G., Yan, S., Zhang, C.: Learning efficient convolutional networks through network slimming. In: ICCV. pp. 2755–2763. IEEE Computer Society (2017)
24. Liu, Z., Sun, M., Zhou, T., Huang, G., Darrell, T.: Rethinking the value of network pruning. In: ICLR (Poster) (2019), https://openreview.net/forum?id=rJlnB3C5Ym
25. Malach, E., Yehudai, G., Shalev-Shwartz, S., Shamir, O.: Proving the lottery ticket hypothesis: Pruning is all you need. In: ICML. Proceedings of Machine Learning Research, vol. 119, pp. 6682–6691. PMLR (2020), http://proceedings.mlr.press/v119/
26. Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A.A., Veness, J., Bellemare, M.G., Graves, A., Riedmiller, M.A., Fidjeland, A., Ostrovski, G., Petersen, S., Beattie, C., Sadik, A., Antonoglou, I., King, H., Kumaran, D., Wierstra, D., Legg, S., Hassabis, D.: Human-level control through deep reinforcement learning. Nat. **518**(7540), 529–533 (2015)
27. Mocanu, D.C., Mocanu, E., Stone, P., Nguyen, P.H., Gibescu, M.: Scalable training of artificial neural networks with adaptive sparse connectivity inspired by network science. Nature Communications (2018). https://doi.org/10.1038/s41467-018-04316-3
28. Morcos, A.S., Yu, H., Paganini, M., Tian, Y.: One ticket to win them all: generalizing lottery ticket initializations across datasets and optimizers. In: NeurIPS. pp. 4933–4943 (2019), https://proceedings.neurips.cc/paper/2019/hash/a4613e8d72a61b3b69b32d040f89ad81-Abstract.html

29. Orseau, L., Hutter, M., Rivasplata, O.: Logarithmic pruning is all you need. In: NeurIPS (2020), https://proceedings.neurips.cc/paper/2020/hash/1e9491470749d5b0e361ce4f0b24d037-Abstract.html
30. Qu, M., Tang, J., Han, J.: Curriculum learning for heterogeneous star network embedding via deep reinforcement learning. In: WSDM. pp. 468–476. ACM (2018)
31. Real, E., Moore, S., Selle, A., Saxena, S., Suematsu, Y.L., Tan, J., Le, Q.V., Kurakin, A.: Large-scale evolution of image classifiers. In: ICML. Proceedings of Machine Learning Research, vol. 70, pp. 2902–2911. PMLR (2017)
32. Su, J., Chen, Y., Cai, T., Wu, T., Gao, R., Wang, L., Lee, J.D.: Sanity-checking pruning methods: Random tickets can win the jackpot. In: NeurIPS (2020)
33. Tanaka, H., Kunin, D., Yamins, D.L., Ganguli, S.: Pruning neural networks without any data by iteratively conserving synaptic flow. In: NeurIPS (2020), https://proceedings.neurips.cc/paper/2020/hash/46a4378f835dc8040c8057beb6a2da52-Abstract.html
34. Vysogorets, A., Kempe, J.: Connectivity matters: Neural network pruning through the lens of effective sparsity (2021). https://doi.org/10.48550/ARXIV.2107.02306
35. Wang, C., Zhang, G., Grosse, R.B.: Picking winning tickets before training by preserving gradient flow. In: ICLR (2020), https://openreview.net/forum?id=SkgsACVKPH
36. Wen, W., Wu, C., Wang, Y., Chen, Y., Li, H.: Learning structured sparsity in deep neural networks. In: NIPS. pp. 2074–2082 (2016)
37. Zhu, M., Gupta, S.: To prune, or not to prune: Exploring the efficacy of pruning for model compression. In: ICLR (Workshop). OpenReview.net (2018)
38. Zoph, B., Le, Q.V.: Neural architecture search with reinforcement learning. In: ICLR. OpenReview.net (2017)
39. Zoph, B., Vasudevan, V., Shlens, J., Le, Q.V.: Learning transferable architectures for scalable image recognition. In: CVPR. pp. 8697–8710. Computer Vision Foundation / IEEE Computer Society (2018)
40. Zou, Z., Shi, Z., Guo, Y., Ye, J.: Object detection in 20 years: A survey. CoRR **abs/1905.05055** (2019)