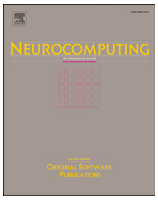
Neurocomputing 661 (2026) 131885

Contents lists available at ScienceDirect

Neurocomputing

journal homepage: www.elsevier.com/locate/neucom



Automatic data-free pruning via channel similarity reconstruction

Siqi Li [a], Jun Chen [b,c,*], Jingyang Xiang [a], Chengrui Zhu [a], Jiandang Yang [a], Xiaobin Wei [d], Yunliang Jiang [c,e], Yong Liu [a,**]

[a] Institute of Cyber-Systems and Control, Zhejiang University, Hangzhou, 310027, China
[b] National Special Education Resource Center for Children with Autism, Zhejiang Normal University, Hangzhou, 311231, China
[c] School of Computer Science and Technology, Zhejiang Normal University, Jinhua, 321004, China
[d] Wasu Media & Network CO. Ltd, Hangzhou, China
[e] School of Information Engineering, Huzhou University, Huzhou, 313000, China

ARTICLE INFO

Communicated by T. Chen

Keywords:
Deep neural networks
Model compression
Network pruning
Data-free pruning

ABSTRACT

Structured pruning methods are developed to bridge the gap between the massive scale of neural networks and the limited hardware resources. Most current structured pruning methods rely on training datasets to fine-tune the compressed model, resulting in high computational burdens and being inapplicable for scenarios with stringent requirements on privacy and security. As an alternative, some data-free methods have been proposed, however, these methods often require handcrafted parameter tuning and can only achieve inflexible reconstruction. In this paper, we propose the Automatic Data-Free Pruning (AutoDFP) method that achieves automatic pruning and reconstruction without fine-tuning. Our approach is based on the assumption that the loss of information can be partially compensated by retaining focused information from similar channels. Specifically, we formulate data-free pruning as an optimization problem, which can be effectively addressed through reinforcement learning. AutoDFP assesses the similarity of channels for each layer and provides this information to the reinforcement learning agent, guiding the pruning and reconstruction process of the network. We evaluate AutoDFP with multiple networks on multiple datasets, achieving impressive compression results.

1. Introduction

The field of model compression has undergone extensive research to bridge the gap between the size of neural networks and the hardware limitations of edge devices, such as pruning [1–5], quantization [6–9], and knowledge distillation [10–12]. Model compression technology known as network pruning aims to reduce redundant parameters and computations in the original networks. Pruning algorithms can be categorized into fine-grained pruning and structured pruning depending on the level of granularity. Since fine-grained pruning [13] produces irregular sparse patterns and requires specialized hardware support [14], many studies have focused on structured pruning. Numerous structured pruning methods have been proposed [15–19]. While these methods have achieved outstanding performance, they require original training data to restore accuracy through fine-tuning or knowledge distillation, being both data-dependent and computationally expensive. In situations where data privacy and security are critical, such as real-world applications involving restricted datasets like medical data and user data, these data-driven methods become unsuitable to employ. Consequently, multiple methods are proposed to perform data-free pruning [20–23]. Data-free pruning techniques, such as the ones presented in DeepInversion [20] and data-free network pruning (DFNP) [21], utilize synthetic samples. Although these methods achieve pruning without the need for data, the process of generating synthetic data is computationally intensive and expensive. Another category of data-free pruning [22–24] proposes compensation for the pruned portion to restore the accuracy of the network. However, these approaches demand significant human efforts to modify pruning strategies and hyperparameters. Additionally, they are limited in their ability to identify network redundancies and provide inflexible compensation for pruned channels, resulting in a significant drop in accuracy. Taking the Neuron Merging (NM) [22] as an example, on the ImageNet dataset, pruning 30 % of the parameters in the ResNet-50 network, results in top-1 accuracy rate of a mere 24.63 %, which is unacceptable. While data-free pruning methods

* Corresponding author at: National Special Education Resource Center for Children with Autism, Zhejiang Normal University, Hangzhou, 311231, China.
** Corresponding author.
Email addresses: junc.change@zjnu.edu.cn (J. Chen), yongliu@iipc.zju.edu.cn (Y. Liu).

https://doi.org/10.1016/j.neucom.2025.131885
Received 23 October 2024; Received in revised form 23 September 2025; Accepted 20 October 2025
Available online 27 October 2025

may not always match the final accuracy of data-driven pruning with fine-tuning, their substantially faster recovery time makes them highly attractive for deployment in hardware-accelerated or privacy-sensitive scenarios.

To address this problem, we aim to automate the process of pruning and compensation based on the redundancy characteristics of each layer. Illustrated in Fig. 2, T-distributed stochastic neighbor embedding (T-SNE) is employed to visualize the clustering results of a particular layer of channels across multiple networks utilizing Density-Based Spatial Clustering of Applications with Noise (DBSCAN) [25]. Through channel clustering, we discovered that they share similarities, which can also be seen as a certain degree of redundancy in the networks. Based on this observation, we hypothesize that when channels are pruned, the loss of information can be partially compensated for by preserving information from similar channels. Therefore, we propose the Automatic Data-Free Pruning (AutoDFP) method, which autonomously evaluates network redundancy levels and devises corresponding pruning and reconstruction strategies. An overview of our AutoDFP approach is depicted in Fig. 1. By incorporating reinforcement learning, we can automatically derive the optimal strategy within the reinforcement learning framework, as depicted in Fig. 3. The main contributions are summarized as follows:

- We formulate data-free pruning and reconstruction as an optimization problem based on the assumption of channel similarity. We model the resolution process of this pruning-reconstruction optimization problem as a Markov decision process, enabling its solution through reinforcement learning algorithms.
- We employ a Soft Actor-Critic (SAC) [26] agent to automate the process of pruning and reconstruction. The SAC agent receives the state of each layer in the network and provides strategies for both pruning and reconstruction.
- The broad applicability of our method is demonstrated across a range of deep neural networks (DNNs), including VGG-16/19 [27], MobileNet-V1 [28]/V2 [29], ResNet-56/34/50/101 [30], and various datasets such as CIFAR-10 and ImageNet. Additionally, we have also evaluated our method on detection networks.

2. Related works

2.1. Automatic network pruning

Numerous methods have been proposed to achieve automated model compression, especially in the field of network pruning [15,16,18,19,31–33].

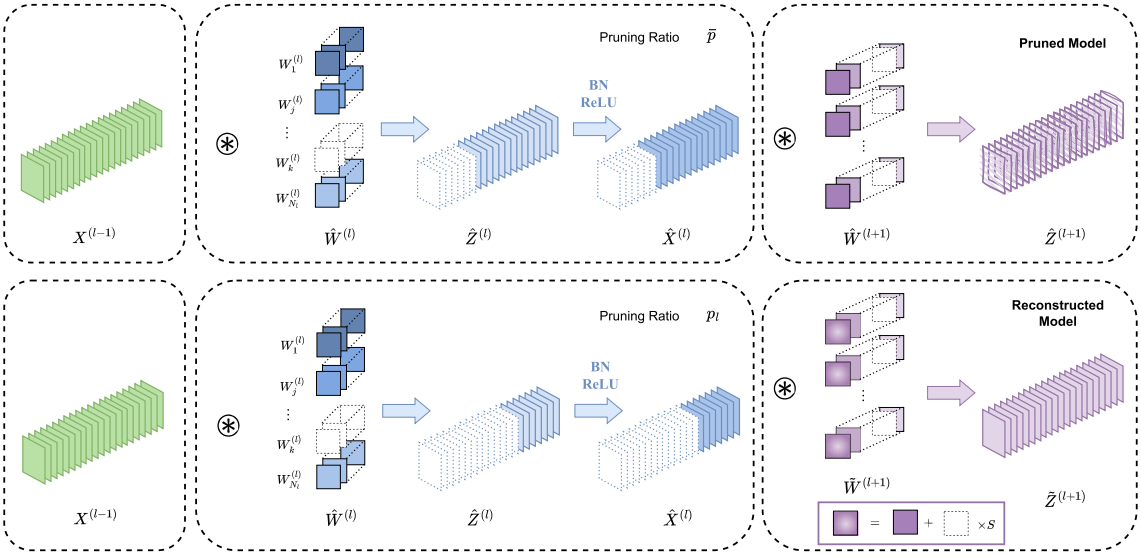


Fig. 1. The overview of AutoDFP. The upper part of the figure displays the outcome of pruning the ℓ^{th} layer using the conventional method with the constant pruning rate $\bar{p}$. As a result, the $(\ell+1)^{th}$ layer acquires a damaged feature map $\hat{Z}^{(\ell+1)}$. The bottom part of the figure demonstrates the procedure of pruning the ℓ^{th} layer and reconstructing the $(\ell+1)^{th}$ layer with the AutoDFP method. Both the specially designed pruning ratio p_l and the reconstruction within the purple box are guided by a reinforcement learning agent, which ultimately generates the restored feature map $\tilde{Z}^{(\ell+1)}$.

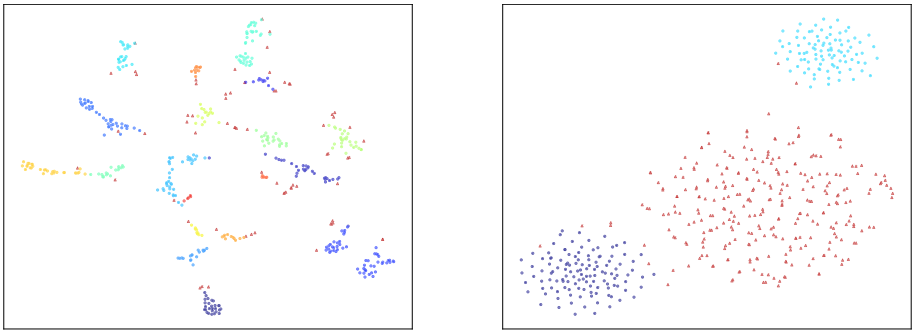


Fig. 2. T-SNE visualization of the results of DBSCAN clustering of channels in a certain layer of the network, the points of different colors represent different clusters. Left: The clustering result of channels of a certain layer in the VGG-16, the red points represent noise points. Right: The clustering result of a certain layer in the ResNet-101.

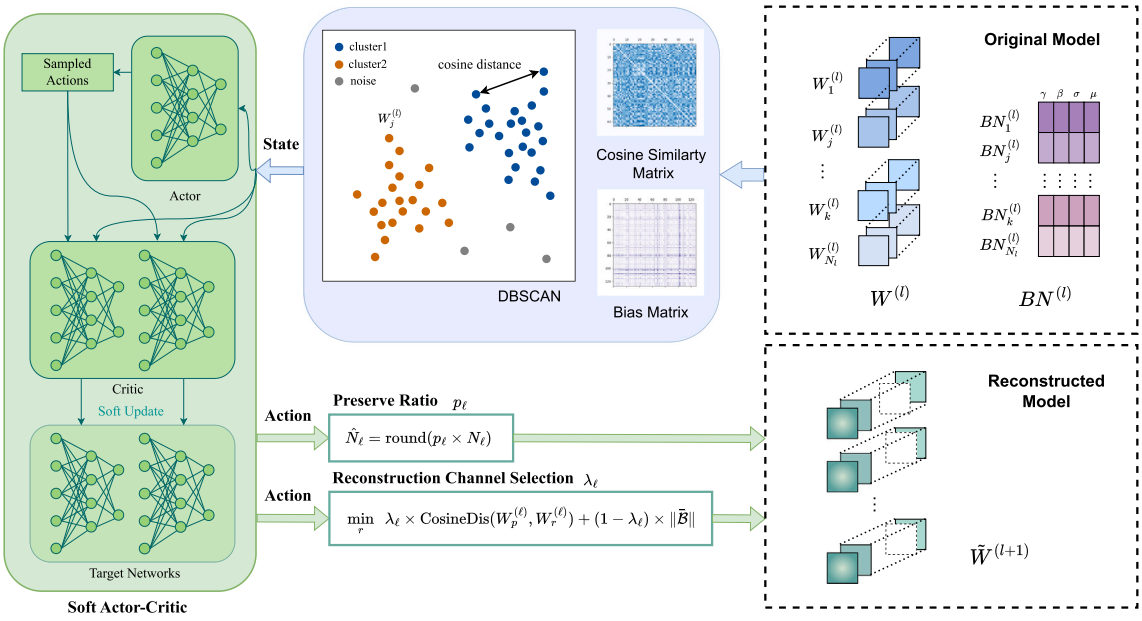


Fig. 3. The framework on which reinforcement learning works in AutoDFP. As depicted in the blue box, AutoDFP utilizes the DBSCAN clustering algorithm and the bias matrix to perform the channel similarity evaluation for each layer of the original model. The state containing the aforementioned information will be furnished to a Soft Actor-Critic agent, which will subsequently produce two continuous actions, namely p_ℓ and λ_ℓ. These actions are then utilized to direct the network's pruning and reconstruction procedures.

Network-to-network compression (N2N) learning [15] determines the strategy of network pruning via policy gradient reinforcement learning. In AutoML for model compression (AMC) [16], a Deep Deterministic Policy Gradient (DDPG) agent is utilized to identify the optimal pruning strategy. Deep Q-Learning for Pruning (QLP) [33] proposes using deep Q-learning to achieve unstructured progressive pruning. Deep compression with reinforcement learning (DECORE) [31] employs a multi-agent reinforcement learning technique to determine the necessity of pruning each channel. Additionally, reinforcement learning and Monte Carlo tree search (RL-MCTS) [32] proposes a method that integrates Monte Carlo tree search into reinforcement learning training to enhance sample efficiency, resulting in improved filter selection. Auto Graph encoder-decoder Model Compression (AGMC) [18] and Graph Neural Networks with Reinforcement Learning (GNN-RL) [19] represent the DNN as a graph, apply graph neural networks (GNNs) to capture its features, and then leverage reinforcement learning to search for effective pruning strategies.

While the aforementioned methods enable automatic network pruning, they rely on access to data for tasks like fine-tuning or knowledge distillation in order to restore network accuracy. Due to the high cost of fine-tuning and potential privacy issues, data-driven automatic compression methods may not be feasible in certain situations.

2.2. Data-free pruning

Some pruning methods attempt to use less fine-tuning and less dependency on the dataset [34–37]. ThiNet [34] models channel pruning as an optimization problem to minimize layer-wise reconstruction error and proposes a solution that does not require using the entire dataset. The approach proposed by He et al. [35] suggests a method for channel selection based on Least Absolute Shrinkage and Selection Operator (LASSO) regression, followed by a reconstruction of output feature maps using least squares. Another method proposed by Mussay et al. [36] introduces a pruning criterion based on coreset to perform channel selection in order to reduce the expensive cost of fine-tuning. Moreover, some data-free approaches utilize synthetic samples to fine-tune the pruned model, such as DeepInversion [20] and DFNP [21]. Although these generative techniques tackle data security concerns, they still necessitate expensive fine-tuning, which can be even more costly.

There exist only a handful of methods capable of achieving network pruning entirely without the need for data or fine-tuning [22–24]. The method proposed by Srinivas et al. [23] first introduces the data-free methods to remove the redundant neurons. Neuron Merging [22] aims to substitute pruned channels with similar ones. In contrast, Data Flow driven Pruning of Coupled Channels (DFPC) [24] primarily focuses on the exploration of coupled channel pruning in data-free scenarios. However, significant manual effort is required to adjust the pruning strategy and hyperparameters in this approach, and the resultant decrease in accuracy is substantial, making it incomparable to the accuracy achieved by the data-driven methods.

3. Problem formulation

3.1. Background and notation

Consider a convolutional neural network (CNN) consisting of L layers. The feature map of the ℓ^{th} layer can be denoted as:

$$\begin{cases} Z^{(\ell)} = X^{(\ell-1)} \circledast \mathcal{W}^{(\ell)} \\ X^{(\ell)} = \mathcal{A}(\mathcal{BN}(Z^{(\ell)})) \end{cases} \tag{1}$$

where $\circledast$ denotes the convolution operation, $\mathcal{BN}(\cdot)$ represents the batch normalization and $\mathcal{A}(\cdot)$ denotes the activation function. $\mathcal{W}^{(\ell)}$, $X^{(\ell-1)}$, $X^{(\ell)}$ and $Z^{(\ell)}$ are tensors. $\mathcal{W}^{(\ell)} \in \mathbb{R}^{N_\ell \times N_{\ell-1} \times K_\ell \times K_\ell}$ represents the weights of the convolutional layer, where N_ℓ is the output channel, $N_{\ell-1}$ is the input channel and $K_\ell \times K_\ell$ is the kernel size. $X^{(\ell-1)} \in \mathbb{R}^{N_{\ell-1} \times h_{\ell-1} \times w_{\ell-1}}$ and $X^{(\ell)} \in \mathbb{R}^{N_\ell \times h_\ell \times w_\ell}$ are the feature maps, where $N_{\ell-1}$ and N_ℓ respectively represent the number of channels. Without taking into account the batch normalization layer and activation function, we denote the feature map as $Z^{(\ell)}$, where $Z^{(\ell)} \in \mathbb{R}^{N_\ell \times h_\ell \times w_\ell}$, which has the same shape as $X^{(\ell)}$.

Next, consider the pruning of the output channels of the ℓ^{th} layer. After pruning, the weights are represented by $\hat{\mathcal{W}}^{(\ell)}$, which has dimensions $\hat{\mathcal{W}}^{(\ell)} \in \mathbb{R}^{\hat{N}_\ell \times N_{\ell-1} \times K_\ell \times K_\ell}$. Here, $\hat{N}_\ell$ denotes the number of remaining output channels after pruning. Similarly, $\hat{X}^{(\ell)}$ represents the feature map obtained after pruning, with dimensions $\hat{X}^{(\ell)} \in \mathbb{R}^{\hat{N}_\ell \times h_\ell \times w_\ell}$.

The pruning of $\hat{X}^{(\ell)}$ leads to modifications in the feature map of the subsequent layer, as follows:

$$\begin{cases} \hat{Z}^{(\ell+1)} = \hat{X}^{(\ell)} \circledast \hat{\mathcal{W}}^{(\ell+1)} \not\approx Z^{(\ell+1)} \\ \hat{X}^{(\ell+1)} = \mathcal{A}(\mathcal{BN}(\hat{Z}^{(\ell+1)})) \not\approx X^{(\ell+1)} \end{cases} \tag{2}$$

$\hat{X}^{(\ell+1)}$ and $\hat{Z}^{(\ell+1)}$ represent the corrupted feature map. $\hat{\mathcal{W}}^{(\ell+1)}$ represents the damaged weights of the $(\ell+1)^{th}$ layer, where the pruned input channels align with the pruned output channels of $\hat{\mathcal{W}}^{(\ell)}$. It has dimensions $\hat{\mathcal{W}}^{(\ell+1)} \in \mathbb{R}^{N_{\ell+1} \times \hat{N}_{\ell} \times K_{\ell+1} \times K_{\ell+1}}$.

3.2. Reconstruction assumption

As depicted in Fig. 2, we notice that certain channels within a neural network demonstrate resemblance, indicating the presence of some redundancy. Based on this observation, we draw upon Neuron Merging [22] and put forth an assumption that the information lost due to the pruning of a weight channel in the $(\ell+1)^{th}$-layer kernel, denoted as $W_p^{(\ell+1)}$ can be partly compensated for by the information present in one of the remaining, similar weight channels $W_r^{(\ell+1)}$:

$$\tilde{W}_r^{(\ell+1)} = W_r^{(\ell+1)} + s_{pr} W_p^{(\ell+1)} \tag{3}$$

where s_{pr} is the reconstruction scale, and $W_r^{(\ell+1)}, W_p^{(\ell+1)}, \tilde{W}_r^{(\ell+1)} \in \mathbb{R}^{(N_{\ell+1} \times K_{\ell+1} \times K_{\ell+1})}$ represent the vectorized kernels associated with specific weight channels (different from the full tensor $\mathcal{W}$).

After reconstruction of the $(\ell+1)^{th}$ layer, the feature map of this layer can be represented as:

$$\begin{cases} \tilde{Z}^{(\ell+1)} = \hat{X}^{(\ell)} \circledast \tilde{\mathcal{W}}^{(\ell+1)} \approx Z^{(\ell+1)} \\ \tilde{X}^{(\ell+1)} = \mathcal{A}(\mathcal{BN}(\tilde{Z}^{(\ell+1)})) \approx X^{(\ell+1)} \end{cases} \tag{4}$$

Based on our assumption, the information loss caused by the pruning of the output feature channels in the ℓ^{th} layer—which serve as the input feature channels of the $(\ell+1)^{th}$ layer—can be compensated by reconstructing the feature map in the $(\ell+1)^{th}$ layer.

3.3. Problem definition

Based on the aforementioned considerations, the delivery matrix S_ℓ and the number of preserved channels $\hat{N}_\ell$ are crucial for data-free pruning.

The size of $\hat{N}_\ell$, determined by the pruning ratio, is directly linked to the degree of information loss incurred during pruning. Additionally, the configuration of S_ℓ plays a crucial role in information reconstruction and recovery, where S_ℓ is the matrix form of s_{pr}. Considering the changes in the quantity of information and redundancy in each layer of the convolutional neural network, selecting suitable values of $\hat{N}_\ell$ and S_ℓ for every prunable layer are desirable. Thus, for each prunable layer ℓ, the problem can be expressed as follows:

$$\min_{\hat{N}_\ell, S_\ell} \|Z^{(\ell+1)} - \tilde{Z}^{(\ell+1)}\| \tag{5}$$

Our goal, when the pruning rate for the entire network is predetermined, is to determine the best values for $\hat{N}_\ell$ and matrix S_ℓ for each prunable layer. The objective is to minimize the overall loss of information in the feature maps of the subsequent layer, ultimately leading to a reduction in the performance loss of the network.

4. Methodology

In this section, we elaborate on our approach. Section 4.1 provides the derivation and proof of layer-wise reconstruction, ultimately modeling it as a multi-objective optimization problem. In Section 4.2, we explain how the layer-by-layer pruning-reconstruction process can be modeled as a Markov Decision Process (MDP). Then, in Section 4.3, we propose using reinforcement learning to solve this Markov Decision problem and provide a specific design for the reinforcement learning agent.

4.1. Layer-wise reconstruction

For layer-wise reconstruction, we adapt and improve upon the method from Neuron Merging [22]. Specifically, for the $(l+1)^{th}$ layer, we consider the scenario where one channel is pruned. When pruning the p^{th} input channel and compensating with the r^{th} channel, the reconstructed feature map for the i^{th} output channel is:

$$\tilde{\mathbf{Z}}_i^{(\ell+1)} = \sum_{j=1, j\neq p}^{N_\ell} X_j^{(\ell)} \circledast W_{i,j}^{(\ell+1)} + X_r^{(\ell)} \circledast s_{pr} W_{i,p}^{(\ell+1)} \tag{6}$$

The reconstruction error is defined as:

$$error = \sum_{i=1}^{N_{\ell+1}} \left\| \mathbf{Z}_i^{(\ell+1)} - \tilde{\mathbf{Z}}_i^{(\ell+1)} \right\| = \sum_{i=1}^{N_{\ell+1}} \left\| \left(X_p^{(\ell)} - s_{pr} X_r^{(\ell)} \right) \circledast W_{i,p}^{(\ell+1)} \right\| \tag{7}$$

Our objective is to minimize this reconstruction error. By simplifying, we reduce this to the following optimization problem:

$$\min \left\| X_p^{(\ell)} - s_{pr} X_r^{(\ell)} \right\| \tag{8}$$

When the activation function is given, we consider the following optimization problem:

$$\min \left\| \mathcal{BN}(Z_p^{(\ell)}) - s_{pr} \mathcal{BN}(Z_r^{(\ell)}) \right\| \tag{9}$$

where $\mathcal{BN}(Z_j) = \gamma_j \frac{Z_j - \mu_j}{\sigma_j} + \beta_j$ and $Z^{(\ell)} = X^{(\ell-1)} \circledast W^{(\ell)}$. As we cannot obtain $X^{(\ell-1)}$ without data, we rely on the model's pre-trained information to solve the optimization problem. Therefore, we minimize:

$$\min(\|\mathcal{E}\|, \|\mathcal{B}\|) \tag{10}$$

where $\mathcal{E}$ and $\mathcal{B}$ are defined as:

$$\begin{cases} \mathcal{E} = W_p^{(\ell)} - s_{pr} \dfrac{\gamma_r}{\sigma_r} \dfrac{\sigma_p}{\gamma_p} W_r^{(\ell)} \\ \mathcal{B} = s_{pr} \left(\dfrac{\gamma_r}{\sigma_r} \mu_r - \beta_r \right) - \dfrac{\gamma_p}{\sigma_p} \mu_p + \beta_p \end{cases} \tag{11}$$

To minimize $\|\mathcal{E}\|$, we calculate the scalar s_{pr} as:

$$s_{pr} = \frac{\|W_p^{(\ell)}\|}{\|W_r^{(\ell)}\|} \cdot \frac{\sigma_r}{\gamma_r} \cdot \frac{\gamma_p}{\sigma_p} \tag{12}$$

This reduces $\mathcal{E}$ to $W_p^{(\ell)} - \frac{\|W_p^{(\ell)}\|}{\|W_r^{(\ell)}\|} W_r^{(\ell)}$, linking error minimization to cosine distance:

$$\min_r \left(\text{CosineDis} \left(W_p^{(\ell)}, W_r^{(\ell)} \right), \|\mathcal{B}\| \right). \tag{13}$$

The framework extends to multi-channel pruning by iteratively applying this compensation strategy, maintaining computational efficiency while preserving layer-wise structural integrity.

4.2. Markov decision process

Given the diverse characteristics of redundancy and weight similarity across different layers in a neural network, it is crucial to determine the optimal pruning ratio and reconstruction channel selection for each layer individually. However, the search space for selecting these parameters can be extensive, making manual selection labor-intensive and prone to suboptimal outcomes. Hence, we propose to utilize reinforcement learning algorithms to address this issue.

The premise for addressing the aforementioned issues using reinforcement learning methods is that our proposed layer-wise pruning-reconstruction optimization problem (as described in Eq. (5)) can be

modeled as a Markov decision process. In the automatic channel pruning and reconstruction process, we leverage the features of each prunable layer $Layer_\ell$ to define the state s_ℓ. The state s_ℓ of each layer is solely determined by the information within that layer. The action a_ℓ is defined as the selection of sparsity and the reconstruction coefficient for layer $Layer_\ell$. Once layer $Layer_\ell$ has been pruned and reconstructed with a_ℓ, the agent moves on to the next layer $Layer_{\ell+1}$ and obtains the subsequent state $s_{\ell+1}$. The validation set is used to determine the reward r after all layers $Layer_L$ have been pruned. It can be observed that the future behavior of the process is dependent only on its current state and does not rely on its past states. This characteristic aligns with the Markov property. It can be inferred that the transition probability for an arbitrary sequence of states $s_0, a_0, s_1, a_1, \ldots, s_{t-1}, a_{t-1}, s_t$ satisfies the Markov property:

$$P(s_t|s_0, a_0, s_1, a_1, \ldots, s_{t-1}, a_{t-1}) = P(s_t|s_{t-1}, a_{t-1}) \tag{14}$$

As a result, the layer-by-layer pruning and reconstruction process can be represented as a Markov Decision Process (MDP), allowing for a reinforcement learning-based solution to the problem.

4.3. Solution via reinforcement learning

As the layer-by-layer pruning and reconstruction process adheres to the Markov property, we utilize a reinforcement learning agent to automatically obtain the pruning strategy and channel reconstruction strategy for each layer. The approach we propose employs a Soft Actor-Critic [26] agent to facilitate the exploration of the extensive search space. As an extension of the Actor-Critic method, SAC merges the concepts of maximum entropy reinforcement learning with the traditional Actor-Critic structure. By incorporating an entropy-based component into the reward function, SAC leverages maximum entropy reinforcement learning principles to promote exploration. The update and search process of our reinforcement learning algorithm is shown in Algorithm 1. Additionally, we design the reward function, action space, and state space for reinforcement learning.

4.3.1. Reward function

The reward function is defined as $r = acc$, where the acc denotes the accuracy evaluated on the validation set after the network has been fully pruned and reconstructed.

4.3.2. Action space

Consider two factors that need to be determined for each layer, namely the number of preserved channels $\hat{N}_\ell$ after pruning and the delivery matrix S_ℓ used during reconstruction. Following this, the action taken by our reinforcement learning agent is represented by a vector $\vec{a}_\ell$, with a defined as $\vec{a}_\ell = [p_\ell, \lambda_\ell]$, which includes the aforementioned two factors respectively.

The first component of the action is the pruning rate p_ℓ for each layer, which symbolizes the degree of information loss. We utilize p_ℓ as a continuous value in the range of $(0, 1]$ and apply the $l2$-norm for pruning. For the ℓ^{th} layer, the number of preserved channels $\hat{N}_\ell$ can be obtained through the p_ℓ:

$$\hat{N}_\ell = \text{round}(p_\ell \times N_\ell) \tag{15}$$

Another component is the coefficient λ_ℓ, which plays a role in channel selection for reconstruction. Since the optimization problem in Eq. (13) constitutes a multi-objective optimization problem and is difficult to solve directly, we aim to find a coefficient $\lambda_\ell \in [0, 1]$ for each layer such that Eq. (13) can be transformed into:

$$\min_r \quad \lambda_\ell \times \text{CosineDis}(W_p^{(\ell)}, W_r^{(\ell)}) + (1 - \lambda_\ell) \times \|\tilde{B}\| \tag{16}$$

where $\|\tilde{B}\|$ is the result of $\|B\|$ in Eq. (11) being normalized in the range of $(0, 1]$. The coefficient λ_ℓ allows us to find the optimal balance between the cosine distance and the bias, enabling us to effectively manage the trade-off between these two values. For each pruned channel $W_p^{(\ell)}$ in the current layer, if selecting a channel $W_r^{(\ell)}$ from the preserved channels satisfies the optimization problem described in Eq. (16), it implies that the optimization problem defined in Eq. (5) is also fulfilled.

Algorithm 1 AutoDFP.

Input: The preservation ratio p_r, a CNN prepared for pruning with the prunable layers $\ell = \{1, \ldots, L\}$, the left bound p_{min} and the right bound p_{max} of the preservation ratio of each layer.
Initial: Initialize the pruning environment, Soft Actor-Critic agent with the policy parameters θ, Q-function parameters ϕ_1, ϕ_2, target network parameters ϕ'_1, ϕ'_2 and an empty replay buffer D_τ.

```
for episode = 1, ..., M do
  for the prunable layer ℓ = 1, ..., L do
    Observe state s and select action a ~ π_θ(·|s), which a = [p_ℓ, λ_ℓ].
    W_other ← Σ_{k=ℓ+1}^{L} p_min W_k − Σ_{k=1}^{ℓ−1} p_k W_k
    p_ℓ ← min(p_ℓ, (p_r W_all − W_other)/W_ℓ)
    Prune output channels of ℓ^th layer and the unprunable layers
    between ℓ^th and (ℓ+1)^th layer with the preserve ratio p_ℓ.
    Prune input channels of (ℓ+1)^th layer with the preserve ratio
    p_ℓ.
    Reconstruct the weights of the (ℓ+1)^th layer with the λ_ℓ.
    if ℓ = L then
      Observe next state s', where s' = s_ℓ is terminal.
      Receive the reward r = acc and observe done signal d = 1.
    else
      Observe next state s', where
      s' = s_{ℓ+1} = (ℓ+1, type, N_ℓ, N_{ℓ+1}, B_mean, P_{B<t}, C_num, C_noise, C_score)
      Receive the reward r = 0 and observe done signal d = 0.
    end if
    Store transition(s, a, r, s', d) in D_τ.
    if s' is terminal then
      Reset environment state.
    end if
    if update then
      Sample a batch of transitions B = {(s, a, r, s', d)} ~ D_τ
      Update Q-function parameters φ_i ← φ_i − λ_Q ∇̂_{φ_i} J_Q(φ_i), i = 1, 2.
      Update policy parameters θ ← θ − λ_π ∇̂_θ J_π(θ).
      Adjust entropy coefficient α ← α − λ ∇̂_α J(α)
      Update target network parameters φ'_i ← τφ_i + (1−τ)φ'_i, i = 1, 2.
    end if
  end for
end for
```

4.3.3. State space

We use 9 features to describe the state s_ℓ for every layer ℓ:

$$s_\ell = (\ell, type, N_{\ell-1}, N_\ell, B_{mean}, \mathcal{P}_{B<t}, C_{num}, C_{noise}, C_{score}) \tag{17}$$

where ℓ is the index of the layer, $type$ is the layer type (the convolutional layer or the fully-connected layer), $N_{\ell-1}$ is the number of input channels and N_ℓ is the number of output channels. B_{mean} represents the mean value of all elements within matrix B, while $\mathcal{P}_{B<t}$ denotes the proportion of elements in B that are less than the threshold t. $B \subset \mathbb{R}^{n \times n}$ is the bias matrix in which each element b_{ij} is determined according to Eq. (11) and Eq. (12):

$$b_{ij} = \left(\frac{\|W_i^{(\ell)}\|}{\|W_j^{(\ell)}\|} \frac{\sigma_j}{\gamma_j} \frac{\gamma_i}{\sigma_i} \right) \cdot \left(\frac{\gamma_j}{\sigma_j} \mu_j - \beta_j \right) - \frac{\gamma_i}{\sigma_i} \mu_i + \beta_i \tag{18}$$

Furthermore, we employ the DBSCAN [25] algorithm to perform clustering on the output channels of the weights based on the cosine

distance. DBSCAN is a density-based spatial clustering algorithm used to identify clusters of arbitrary shapes in a dataset, as well as noise data. C_{num} is the number of clusters (excluding noise points), C_{noise} represents the proportion of noise present in the clustering results, and C_{score} is the silhouette score of the clustering results, which is employed to evaluate the effectiveness of the clustering. Due to its ability to handle an unspecified number of clusters and accommodate noise points that do not belong to any cluster, the DBSCAN algorithm is well-suited for our objective of gauging the similarity of channels within each layer of the network.

4.3.4. SAC agent

During our pruning process, the SAC agent underwent a warm-up phase of 200 episodes, followed by a search phase of 4800 episodes. The agent consists of both two Actor networks and two Critic networks, each with two hidden layers of 256 neurons each. The learning rates for both networks are set to 1e-3, and the Adam optimizer is used. The learning rate for the entropy coefficient, Alpha, is set to 3e-4.

To avoid prioritizing short-term rewards excessively, a discount factor of 1 is utilized. Soft updates for the target networks are carried out using the value of τ set to 0.01, allowing for a gradual and stable update process. The batch size for the agent is set to 128. The size of the replay buffer, which stores past experiences, is determined based on the depth of the network being pruned, ensuring sufficient memory capacity for effective learning and exploration. The replay buffer size is defined as $100 \times L$, where L is the total number of layers.

By utilizing the reinforcement learning agent, we explore the optimal pruning ratio p_ℓ and reconstruction channel selection parameter λ_ℓ for every layer in the network. This approach enables us to tackle the optimization problem presented in Eq. (5).

5. Experiments

In this section, we present a detailed experimental evaluation of our method. Section 5.1 provides ablation studies. Then, Section 5.2 analyzes the pruning and reconstruction strategies. Sections 5.3 and 5.5 present the performance of our method on classification and detection tasks, respectively. Section 5.4 compares the Accuracy-Preserved Ratio Pareto curves of various methods. Finally, Section 5.7 demonstrates the search time required by the AutoDFP method and discusses the efficiency of the approach.

In our experiments, the reinforcement learning settings are provided in Section 4.3.4. For the DBSCAN clustering algorithm, we set the distance threshold eps to 1 and the minimum number of points within a neighborhood to 5 on the ResNet-56 model; eps to 0.3 and the minimum number of points to 5 on the VGG series models; and eps to 0.5 and the minimum number of points to 5 for the remaining models. Clustering is performed using the cosine similarity between channels. Regarding the preserved strategy range during the search process, we set the left bound p_{min} to 0.2 and the right bound p_{max} to 1 for all models.

To ensure experimental fairness, our comparative evaluation was strictly confined to data-free pruning methodologies, explicitly excluding comparisons with non-data-free pruning approaches. This comparison includes two distinct categories of data-free pruning methods: (1) Data-free pruning reconstruction methods (e.g., Neuron Merging [22] and DFPC [24]), and (2) Generative-based data-free pruning approaches (exemplified by DFNP [21]).

5.1. Ablation study

Table 1 shows the pruning results of MobileNet-V2 [29] under different pruning strategies and reconstruction methods on the CIFAR-10 dataset, with a baseline accuracy of 85.48 %. “P-R” represents the preserved ratios of network parameters. “RL Pruning” denotes our utilization of the SAC agent solely for exploring network pruning strategies, wherein the state definition for each layer follows the same approach as defined in AMC [16]. In contrast, a uniform pruning strategy is implemented in instances where the “RL pruning” process is not employed. “Reconstruction” signifies the process of compensating the network after determining the pruning strategy, without the guidance of reinforcement learning, similar to the Neuron Merging [22] method. “Ours” represents our proposed AutoDFP method, which employs a reinforcement learning agent to holistically guide both pruning and reconstruction processes concurrently.

Table 1
The ablation experiments of MobileNet-V2 under different pruning strategies and reconstruction methods on the CIFAR-10 dataset.

P-R	RL Pruning	Reconstruction	AutoDFP	Acc(%)
50 %	✗	✗	✗	47.18
	✗	✓	✗	26.81
	✓	✗	✗	61.33
	✓	✓	✗	78.22
		Ours	✓	**84.14**
40 %	✗	✗	✗	30.40
	✗	✓	✗	20.40
	✓	✗	✗	41.76
	✓	✓	✗	56.80
		Ours	✓	**77.71**
30 %	✗	✗	✗	11.76
	✗	✓	✗	11.75
	✓	✗	✗	13.02
	✓	✓	✗	53.57
		Ours	✓	**60.49**

Due to its lightweight structure and the inverted residual module, MobileNet-V2 is highly sensitive to pruning strategies. Table 1 shows that using the reconstruction method under a uniform pruning strategy results in a greater accuracy loss than without reconstruction. This can be attributed to the inflexibility of reconstruction without reinforcement learning guidance, leading to unacceptable results. Additionally, the accuracy obtained by solely employing reinforcement learning pruning methods or solely relying on reconstruction methods is also unsatisfactory.

Despite the notable improvement in network accuracy achieved by employing both the searched strategy and the reconstruction, pruned networks still exhibit considerable losses. Our approach significantly outperforms the standalone reinforcement learning pruning combined with reconstruction, indicating that the efficacy of our approach is not solely attributed to either the search strategy or the reconstruction, but rather to the overall effectiveness of our methodology.

5.2. Strategy

Our attention is not limited solely to accuracy, but also to exploring pruning and reconstruction strategies.

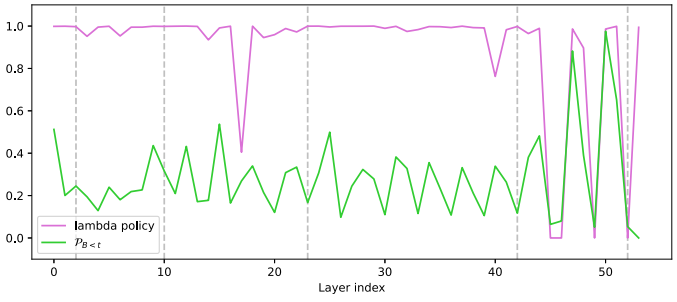


Fig. 4. The value of $\mathcal{P}_{B<t}$ and the reconstruction strategy of ResNet-50 on the ImageNet dataset. The purple line represents the reconstruction strategy λ_ℓ of each layer given by the reinforcement learning agent, and the green line represents a component $\mathcal{P}_{B<t}$ in the state s_ℓ given to the agent.

Fig. 4 showcases the reconstruction strategy of ResNet-50 on the ImageNet dataset. We observed that in the first half of the network, the value of $\mathcal{P}_{B<t}$, representing the proportion of the bias matrix B being less than the threshold t, fluctuates within a relatively narrow range. During this period, the reconstruction strategy provided by the network remains relatively stable. In the latter part of the network, the value of $\mathcal{P}_{B<t}$ undergoes drastic changes. Simultaneously, the reconstruction strategy λ_ℓ also exhibits significant changes consistent with the trend of $\mathcal{P}_{B<t}$. It is evident that during this period, the trends of both variables are perfectly aligned. This indicates that our reconstruction method tends to consider the cosine distance more as a reference for channel selection when the proportion of small values in the bias matrix is relatively high. During this time, the majority of bias values $\|\bar{B}\|$ are very small, there is less need to focus on minimizing them. This is consistent with our understanding as it can effectively address the optimization problem defined in Eq. (13).

Additionally, Fig. 5 demonstrates the pruning strategy of VGG-16 on the CIFAR-10 dataset. It can be observed that the pruning ratios p_ℓ provided by the reinforcement learning agent for each layer correspond to the trend of noise ratio C_{noise} observed in the clustering outcomes. A lower noise rate indicates a higher similarity rate among channels, making reconstruction easier, and thus resulting in a lower proportion of channels being preserved. This supports our hypothesis that if there is significant channel similarity within a particular layer, we are more likely to prune a higher number of channels. Furthermore, this demonstrates that the reinforcement learning method can address the limitations of the channel similarity assumption. This assumption relies on the premise that channels within a given layer exhibit high similarity. However, when the similarity between channels in the layer is low, the assumption becomes less reliable. The reinforcement learning agent can autonomously assess the similarity between channels within the layer and assign a higher preserved rate to layers with low similarity. As a result, fewer or no channels are pruned from these layers, effectively avoids the limitations of the similarity-based compensation approach.

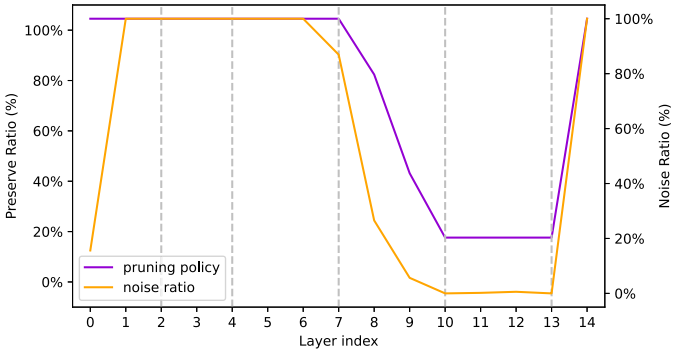


Fig. 5. The value of C_{noise} and the pruning strategy of VGG-16 on the CIFAR-10 dataset. The purple broken line represents the pruning strategy p_ℓ of each layer given by the agent, and the yellow broken line represents a component C_{noise} in the state s_ℓ given to the agent.

Table 3
Pruning results of our method, along with, DFNP [21] and DFPC [24] are compared across VGG-16/19, ResNet-50 on CIFAR-10 datasets. * indicates the method based on the synthetic data. Note that "P-R" denotes the preserved parameter ratio and "F-R" represents the preserved FLOPs ratio.

Model	Method	P-R	F-R	Acc.(%)	ΔAcc.(%)
VGG-16	DFNP* [21]	21.3 %	67.7 %	93.17 → 92.16	−1.01
	Ours	**20.9 %**	**55.4 %**	**93.70 → 92.88**	**−0.82**
VGG-19	DFPC [24]	31.6 %	59.5 %	93.50 → 90.12	−3.38
	DFNP* [21]	23.6 %	65.0 %	93.34 → 92.55	−0.79
	Ours	**23.2 %**	**58.1 %**	**93.90 → 93.39**	**−0.51**
ResNet-50	DFPC [24]	54.9 %	69.4 %	94.99 → 89.95	−5.04
	$DFPC_{CP}$ [24]	48.3 %	68.4 %	94.99 → 90.25	−4.74
	Ours	**37.8 %**	**68.0 %**	**95.00 → 92.42**	**−2.58**

5.3. Classification task

5.3.1. CIFAR-10

Tables 2 and 3 show the pruning results of VGG-16/19 [27], ResNet-56/50 [30] and MobileNet-V1 [28] on the CIFAR-10 datasets. Note that the reported experimental results represent the average outcomes obtained from 5 separate experiments on one NVIDIA GeForce GTX 1080 Ti. We contrast our proposed approach with the data-free pruning method Neuron Merging [22], DFPC [24], and a generative-based data-free pruning method DFNP [21]. Due to the disparate hardware platforms employed in our experiments and those of the comparative study, direct comparison of inference speeds for pruned models proves challenging. Consequently, we employ the floating point operations (FLOPs) preserved ratio as a metric to gauge inference speed.

As shown in Table 2, in comparison to Neuron Merging [22], our method achieves a substantial improvement in accuracy on both ResNet-56 and MobileNet-V1 models with the same preserved ratios of parameters. On the ResNet-56 network, with a parameter preserved ratio of 60 %, our method outperforms Neuron Merging by 8.92 % in accuracy (85.48 % vs. 76.56 %). Meanwhile, on the MobileNet-V1 network, our experiments tested parameter-preserved ratios ranging from 40 % to 20 %. It is worth noting that with such low preserved ratios, our method does not incur any loss in accuracy.

Furthermore, Table 3 presents a comparison of our method with the recent data-free pruning method DFPC [24] and generative-based data-free pruning method DFNP [21]. Across the VGG-16, VGG-19, and ResNet-50 networks, it is evident that under the same or reduced number

Table 2
Pruning results of VGG-16, ResNet-56 and MobileNet-V1 on CIFAR-10 datasets. "P-R" represents the preserved ratios of network parameters.

Model	P-R	Prune	Neuron Merging [22]		Ours	
		Acc.(%)	Acc.(%)	ΔAcc.(%)	Acc.(%)	ΔAcc.(%)
VGG-16 (Acc. 93.70 %)	40 %	89.14	93.16	+4.02	**94.34**	+5.20
	30 %	35.83	65.77	+29.94	**92.94**	+57.11
	20 %	18.15	40.26	+22.11	**90.00**	+71.85
ResNet-56 (Acc. 93.88 %)	70 %	76.95	85.22	+8.27	**88.15**	+11.20
	60 %	46.44	76.56	+30.12	**85.48**	+39.04
	50 %	24.34	56.18	+31.84	**64.29**	+39.95
MobileNet-V1 (Acc. 86.49 %)	40 %	56.92	65.78	+8.86	**86.49**	+29.57
	30 %	31.36	49.87	+18.51	**86.49**	+55.13
	20 %	13.90	37.23	+23.33	**86.50**	+72.60

of FLOPs, our pruning outcomes exhibit comparatively minor accuracy degradation when contrasted with DFNP and DFPC. On the VGG-16 network, our method achieves higher accuracy (-0.76 % vs. -1.01 %) and fewer FLOPs (64.3 % vs. 67.7 %) compared to the DFNP. Meanwhile, on the VGG-19 network, our method achieves pruned networks with fewer FLOPs and parameters, and higher accuracy compared to DFNP and DFPC methods. Notably, particularly on the ResNet-50 network, our approach demonstrates exceptional performance when compared to the DFPC method, regardless of whether this method involves coupled channel pruning (DFPC and DFPC$_{CP}$).

5.3.2. ImageNet

Table 4 shows the pruning results of ResNet-34/50/101 [30] and MobileNet-V1 [28] on the ImageNet dataset with parameter preservation ratios of 90 %, 80 %, and 70 %, respectively. The experiments were conducted using one NVIDIA GeForce RTX 2080 Ti and the results provided are an average of 5 independent experiments.

Our method demonstrated significant improvement in top-1 accuracy compared to the Neuron Merging [22]. Remarkably, when the preserved ratio of ResNet-34 is set to 70 %, our method outperforms the standard pruning method by 41.41 % and the NM method by 19.19 % in terms of top-1 accuracy. Moreover, with the preserved ratio of ResNet-50 set to 70 %, our method demonstrates a notable increase in accuracy, surpassing the standard pruning method by 37.55 % and the NM method by 17.02 %. Likewise, with ResNet-101 maintaining a preserved ratio of 70 %, our method exhibits an accuracy enhancement of 35.39 % compared to the standard pruning method and 8.22 % compared to the NM method. Additionally, our method achieves 43.17 % higher top-1 accuracy than the NM method with the same 80 % preserved ratio on MobileNet-V1.

5.4. Pareto curves

Fig. 6 illustrates the Accuracy-Preserved Ratio Pareto curves of common pruning methods, Neuron Merging [22], and our proposed AutoDFP method across various network structures on multiple datasets. The first row of Fig. 6 shows the Accuracy-Preserved Ratio Pareto curves for VGG-16 [27], ResNet-56 [30], MobileNet-V1 [28], and MobileNet-V2 [29] on the CIFAR-10 dataset. The second row of Fig. 6 displays the Pareto curves of Accuracy-Preserved Ratio for ResNet-34/50/101 [30] on the ImageNet dataset.

It is evident that the Pareto curve of AutoDFP strictly dominates the curves of the other two methods. Particularly in network structures without residual modules, such as VGG-16 and MobileNet-V1, our method demonstrates substantial improvements over the Neuron Merging method.

5.5. Detection task

Besides assessing the AutoDFP method's performance in the classification task, we also conducted experiments in the detection task,

Table 4
Pruning results of ResNet-34, ResNet-50 and ResNet-101 on ImageNet datasets. "P-R" represents the preserved ratios of network parameters.

Model	P-R	Prune	Neuron Merging [22]		Ours	
		Acc.(%)	Acc.(%)	ΔAcc.(%)	Acc.(%)	ΔAcc.(%)
ResNet-34 (Acc. 73.31 %)	90 %	63.71	66.95	+3.24	**70.17**	+6.46
	80 %	42.80	55.54	+12.47	**62.43**	+19.63
	70 %	17.06	39.28	+22.22	**58.47**	+41.41
ResNet-50 (Acc. 76.13 %)	90 %	62.17	68.70	+6.53	**73.77**	+11.60
	80 %	31.35	51.99	+20.64	**60.30**	+28.95
	70 %	4.28	24.63	+20.53	**41.83**	+37.55
ResNet-101 (Acc. 77.31 %)	90 %	68.90	72.29	+3.39	**74.17**	+5.27
	80 %	45.77	61.53	+15.76	**65.19**	+19.42
	70 %	10.34	37.51	+27.17	**45.73**	+35.39
MobileNet-V1 (Acc. 72.03 %)	90 %	15.69	48.45	+32.76	**67.43**	+51.74
	80 %	1.27	15.56	+14.29	**58.73**	+57.46
	70 %	0.52	1.84	+1.32	**44.23**	+43.71

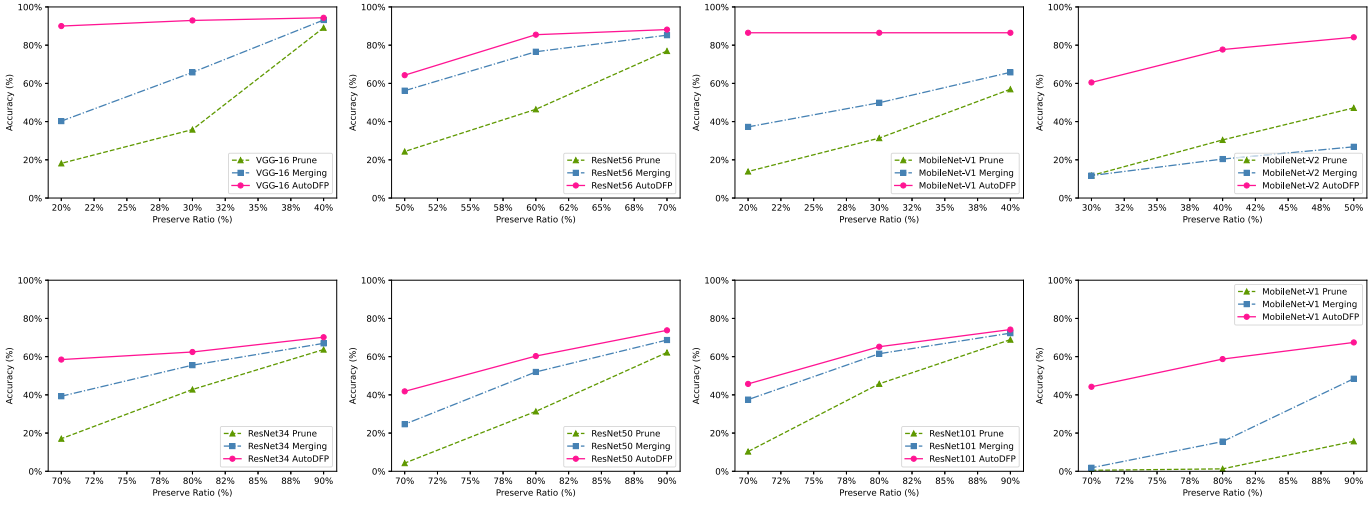


Fig. 6. Comparing the accuracy and preserved ratio trade-off among Prune, Neuron Merging, and AutoDFP on multiple networks.

Table 5
Pruning results of multiple detection networks on COCO2017 dataset when the preserve ratio is set to 90 %.

Model	mAP	
	Baseline	AutoDFP
Faster R-CNN [38]	36.9	35.4
RetinaNet [39]	36.3	35.2
Mask R-CNN [40]	37.8	35.9
FCOS [41]	39.1	37.1

as presented in Table 5. We evaluated various networks, including Faster Region-based Convolutional Neural Network (Faster R-CNN) [38], RetinaNet [39], Mask Region-based Convolutional Neural Network (Mask R-CNN) [40], and Fully Convolutional One-Stage Object detector (FCOS) [41], on the COCO2017 dataset. Note that the reported experimental results represent the average outcomes obtained from 5 separate experiments conducted on a single NVIDIA GeForce RTX 2080 Ti. For the detection networks, we utilize ResNet-50 [30] as the backbone network and apply AutoDFP to prune and reconstruct the backbone network without any subsequent fine-tuning. The baseline model and pre-training weights used in the experiments are obtained from Torchvision [42].

5.6. Search time

We evaluate the efficiency of our proposed AutoDFP method. Specifically, we measure the time and resource consumption for the exploration carried out by the reinforcement learning agent.

Fig. 7 showcases the best reward curves obtained during the search process. The left part of Fig. 7 displays the best reward curves of the MobileNet-V2 network on the CIFAR-10 dataset under different total pruning rates. It can be observed that when the total pruning rate is set to 50 %, the reinforcement learning agent achieves the best rewards around 3000 episodes. For a total pruning rate of 40 %, the agent reaches optimal rewards in just 1215 episodes. The right part of Fig. 7 illustrates the best reward curves of a series of ResNet networks on the ImageNet dataset, with a total preserved ratio set to 70 %. It can be observed that for ResNet-34, ResNet-50, and ResNet-101 networks, the reinforcement learning agent achieves optimal rewards at 1179, 1965, and 2623 episodes, respectively. The results reveal that our reinforcement learning agent is capable of identifying the optimal strategy within 3000 episodes despite the extensive search space.

Furthermore, we measured the GPU hours required for 3000 episodes of search on a single NVIDIA GeForce RTX 2080 Ti, as depicted in Table 6. We compared our time and hardware consumption with the generative-based data-free pruning method DeepInversion [20]. Notably, our time requirements are significantly lower than those of the generative approach, as well as the costs of finetuning ResNet-50 on the ImageNet dataset. For instance, fine-tuning a ResNet-50 model on ImageNet typically requires about 50 GPU hours, while AutoDFP only requires 10.2 GPU hours without fine-tuning, demonstrating a clear advantage in recovery efficiency.

Table 6
Comparison of GPU hours for ResNet-50 on the ImageNet dataset among data-driven pruning with fine-tuning, generative-based data-free pruning (DeepInversion [20]), and our AutoDFP method.

Methods	Hardware	GPU hours
DeepInversion	NVIDIA V100	2800
Data-driven pruning (fine-tuning, 50 epochs)	NVIDIA 2080 Ti	50.6
Ours	NVIDIA 2080 Ti	10.2

Table 7
Inference time (ms) comparison of VGG-16, ResNet-56 and MobileNet-V1 on CIFAR-10 datasets. "P-R" represents the preserved ratios of network parameters. All experiments are conducted on a single NVIDIA GeForce 1080 Ti GPU.

Model	P-R	Prune	Ours
		Times(ms)	Times(ms)
VGG-16	40 %	1.53	**1.44**
	30 %	**1.65**	1.67
	20 %	1.61	**1.54**
ResNet-56	70 %	5.92	**5.62**
	60 %	5.82	**5.26**
	50 %	5.97	**5.65**
MobileNet-V1	40 %	**2.42**	2.54
	30 %	2.49	**2.47**
	20 %	**2.53**	2.64

5.7. Inference time.

We further evaluate the inference latency of the pruned models on a single NVIDIA GeForce 1080Ti GPU, as shown in Table 7. The results indicate that our AutoDFP generally achieves comparable or lower inference time than the baseline pruning method. For example, on VGG-16 and ResNet-56, AutoDFP consistently reduces latency across most preserved ratios (e.g., 1.44ms vs. 1.53ms on VGG-16 with 40 % P-R, 5.26ms vs. 5.82ms on ResNet-56 with 60 % P-R). Although on MobileNet-V1 our method shows slightly higher latency in some cases, the difference remains minor. Considering that AutoDFP also achieves significantly higher accuracy under the same preserved ratios (Table 2), these results demonstrate that AutoDFP offers a better accuracy–efficiency trade-off than competing pruning methods.

6. Conclusion

In this paper, we propose Automatic Data-Free Pruning (AutoDFP), a data-free pruning method designed to automatically provide suitable

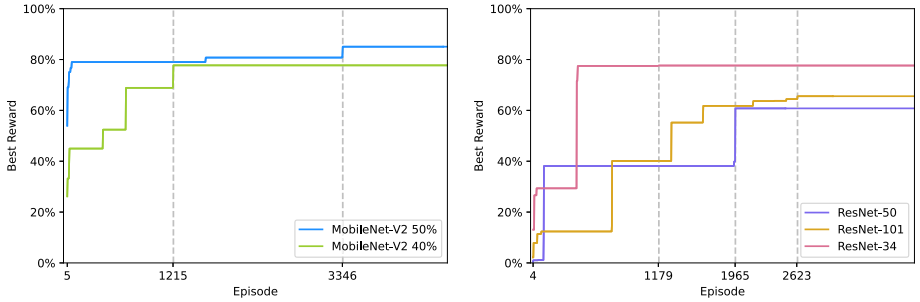


Fig. 7. Left: The best reward curve achieved by the SAC agent on the CIFAR-10 dataset while employing different pruning rates for the MobileNet-V2 network. Right: The best reward curve achieved by the SAC agent on the ImageNet dataset for the ResNet-34/50/101 networks when the preserve ratio is set to 70 %.

pruning and reconstruction guidance for each layer to achieve improved accuracy. We formulate the network pruning and reconstruction task as an optimization problem that can be addressed using a reinforcement learning algorithm. By employing a Soft Actor-Critic agent, we guide the pruning and reconstruction processes in a data-free setting. AutoDFP automatically assesses channel similarity and redundancy at each network layer, facilitating efficient compression and reconstruction. AutoDFP has shown substantial improvements across a wide range of networks and datasets, outperforming the current SOTA method while requiring acceptable search time and computational resources. AutoDFP provides faster recovery than both data-driven pruning (requiring fine-tuning) and generative-based data-free pruning, enabling practical deployment. Furthermore, the pruning and reconstruction strategies derived by AutoDFP are not only reasonable but also explainable, which further supports our approach.

CRediT authorship contribution statement

Siqi Li: Writing – original draft, Software, Resources, Methodology. **Jun Chen:** Supervision. **Jingyang Xiang:** Formal analysis. **Chengrui Zhu:** Formal analysis. **Jiandang Yang:** Funding acquisition. **Xiaobin Wei:** Project administration. **Yunliang Jiang:** Funding acquisition. **Yong Liu:** Supervision.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgement

This research was supported by Zhejiang Provincial Natural Science Foundation of China under Grant No. LQN25F030018.

Data availability

Data will be made available on request.

References

[1] S. Han, J. Pool, J. Tran, W. Dally, Learning both weights and connections for efficient neural network, Adv. Neural Inf. Process. Syst. 28 (2015).
[2] W. Chen, J. Wilson, S. Tyree, K. Weinberger, Y. Chen, Compressing neural networks with the hashing trick, in: International Conference on Machine Learning, PMLR, 2015, pp. 2285–2294.
[3] Z. Liu, J. Li, Z. Shen, G. Huang, S. Yan, C. Zhang, Learning efficient convolutional networks through network slimming, in: Proceedings of the IEEE International Conference on Computer Vision, 2017, pp. 2736–2744.
[4] B. Jiang, J. Chen, Y. Liu, Single-shot pruning and quantization for hardware-friendly neural network acceleration, Eng. Appl. Artif. Intell. 126 (2023) 106816.
[5] C. Zhao, Y. Zhang, B. Ni, Exploiting channel similarity for network pruning, IEEE Trans. Circuits Syst. Video Technol. 33 (2023) 5049–5061, https://doi.org/10.1109/TCSVT.2023.3248659
[6] M. Rastegari, V. Ordonez, J. Redmon, A. Farhadi, Xnor-net: ImageNet classification using binary convolutional neural networks, in: Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, the Netherlands, October 11–14, 2016, Proceedings, Part IV, Springer, 2016, pp. 525–542.
[7] J. Chen, L. Liu, Y. Liu, X. Zeng, A learning framework for n-bit quantized neural networks toward FPGAs, IEEE Trans. Neural Netw. Learn. Syst. 32 (2020) 1067–1081.
[8] F. Li, B. Zhang, B. Liu, Ternary weight networks, arXiv preprint arXiv:1605.04711 (2016).
[9] I. Hubara, M. Courbariaux, D. Soudry, R. El-Yaniv, Y. Bengio, Binarized neural networks, Adv. Neural Inf. Process. Syst. 29 (2016).
[10] G. Hinton, O. Vinyals, J. Dean, Distilling the knowledge in a neural network, Stat 1050 (2015) 9.
[11] Y. Liu, J. Chen, Y. Liu, Dccd: reducing neural network redundancy via distillation, IEEE Trans. Neural Netw. Learn. Syst. (2023).
[12] N. Komodakis, S. Zagoruyko, Paying more attention to attention: improving the performance of convolutional neural networks via attention transfer, in: ICLR, 2017.
[13] S. Han, J. Pool, J. Tran, W. Dally, Learning both weights and connections for efficient neural network, Adv. Neural Inf. Process. Syst. 28 (2015).
[14] S. Han, X. Liu, H. Mao, J. Pu, A. Pedram, M.A. Horowitz, W.J. Dally, Eie: efficient inference engine on compressed deep neural network, ACM SIGARCH Comput. Archit. News 44 (2016) 243–254.
[15] A. Ashok, N. Rhinehart, F. Beainy, K.M. Kitani, N2N learning: network to network compression via policy gradient reinforcement learning, in: International Conference on Learning Representations, 2018.
[16] Y. He, J. Lin, Z. Liu, H. Wang, L.-J. Li, S. Han, Amc: automl for model compression and acceleration on mobile devices, in: European Conference on Computer Vision (ECCV), 2018.
[17] M. Lin, R. Ji, Y. Zhang, B. Zhang, Y. Wu, Y. Tian, Channel pruning via automatic structure search, arXiv preprint arXiv:2001.08565 (2020).
[18] S. Yu, A. Mazaheri, A. Jannesari, Auto graph encoder-decoder for neural network pruning, in: Proceedings of the IEEE/CVF International Conference on Computer Vision, 2021, pp. 6362–6372.
[19] S. Yu, A. Mazaheri, A. Jannesari, Topology-aware network pruning using multi-stage graph embedding and reinforcement learning, in: International Conference on Machine Learning, PMLR, 2022, pp. 25656–25667.
[20] H. Yin, P. Molchanov, J.M. Alvarez, Z. Li, A. Mallya, D. Hoiem, N.K. Jha, J. Kautz, Dreaming to distill: data-free knowledge transfer via deepinversion, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2020, pp. 8715–8724.
[21] J. Tang, M. Liu, N. Jiang, H. Cai, W. Yu, J. Zhou, Data-free network pruning for model compression, in: 2021 IEEE International Symposium on Circuits and Systems (ISCAS), IEEE, 2021, pp. 1–5.
[22] W. Kim, S. Kim, M. Park, G. Jeon, Neuron merging: compensating for pruned neurons, Adv. Neural Inf. Process. Syst. 33 (2020) 585–595.
[23] S. Srinivas, R.V. Babu, Data-free parameter pruning for deep neural networks, arXiv preprint arXiv:1507.06149 (2015).
[24] T. Narshana, C. Murti, C. Bhattacharyya, Dfpc: data flow driven pruning of coupled channels without data, in: The Eleventh International Conference on Learning Representations, 2022.
[25] M. Ester, H.-P. Kriegel, J. Sander, X. Xu, et al., A density-based algorithm for discovering clusters in large spatial databases with noise, in: KDD, vol. 96, 1996, pp. 226–231.
[26] T. Haarnoja, A. Zhou, P. Abbeel, S. Levine, Soft actor-critic: off-policy maximum entropy deep reinforcement learning with a stochastic actor, in: Proceedings of the 35th International Conference on Machine Learning, PMLR, 2018, pp. 1861–1870.
[27] K. Simonyan, A. Zisserman, Very deep convolutional networks for large-scale image recognition, arXiv preprint arXiv:1409.1556 (2014).
[28] A.G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, H. Adam, Mobilenets: efficient convolutional neural networks for mobile vision applications, arXiv preprint arXiv:1704.04861 (2017).
[29] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, L.-C. Chen, Mobilenetv2: inverted residuals and linear bottlenecks, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2018, pp. 4510–4520.
[30] K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016, pp. 770–778.
[31] M. Alwani, Y. Wang, V. Madhavan, Decore: deep compression with reinforcement learning, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2022, pp. 12349–12359.
[32] Z. Wang, C. Li, Channel pruning via lookahead search guided reinforcement learning, in: Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision, 2022, pp. 2029–2040.
[33] E. Camci, M. Gupta, M. Wu, J. Lin, Qlp: deep q-learning for pruning deep neural networks, IEEE Trans. Circuits Syst. Video Technol. 32 (2022) 6488–6501, https://doi.org/10.1109/TCSVT.2022.3167951
[34] J.-H. Luo, J. Wu, W. Lin, Thinet: a filter level pruning method for deep neural network compression, in: Proceedings of the IEEE International Conference on Computer Vision, 2017, pp. 5058–5066.
[35] Y. He, X. Zhang, J. Sun, Channel pruning for accelerating very deep neural networks, in: Proceedings of the IEEE International Conference on Computer Vision, 2017, pp. 1389–1397.
[36] B. Mussay, M. Osadchy, V. Braverman, S. Zhou, D. Feldman, Data-independent neural pruning via coresets, in: International Conference on Learning Representations, 2020.
[37] Y. Tang, S. You, C. Xu, J. Han, C. Qian, B. Shi, C. Xu, C. Zhang, Reborn filters: pruning convolutional neural networks with limited data, in: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 34, 2020, pp. 5972–5980.
[38] S. Ren, K. He, R. Girshick, J. Sun, Faster R-CNN: towards real-time object detection with region proposal networks, Adv. Neural Inf. Process. Syst. 28 (2015).
[39] T.-Y. Lin, P. Goyal, R. Girshick, K. He, P. Dollár, Focal loss for dense object detection, in: Proceedings of the IEEE International Conference on Computer Vision, 2017, pp. 2980–2988.
[40] K. He, G. Gkioxari, P. Dollár, R. Girshick, Mask R-CNN, in: Proceedings of the IEEE International Conference on Computer Vision, 2017, pp. 2961–2969.
[41] Z. Tian, C. Shen, H. Chen, T. He, Fcos: fully convolutional one-stage object detection, in: Proceedings of the IEEE/CVF International Conference on Computer Vision, 2019, pp. 9627–9636.
[42] T. maintainers, contributors, Torchvision: PyTorch's computer vision library (2016) https://github.com/pytorch/vision.

Author biography

Siqi Li received the B.Eng. degree in automation from Xi'an Jiaotong University, Xi'an, China, in 2022, where she is currently pursuing the Ph.D. degree with the Institute of Cyber Systems and Control, Department of Control Science and Engineering, Zhejiang University, Hangzhou, China. Her research interests include neural network compression and deep learning.

Jun Chen received the Ph.D degree in control science and engineering from Zhejiang University, Hangzhou, China, in 2024. He is currently a distinguished professor in Zhejiang Normal University, Jinhua, China. His research interests include deep learning, model compression, decentralized optimization, and manifold optimization.

Jingyang Xiang received the B.S. degree in electrical engineering and automation from the Zhejiang University of Technology, Hangzhou, China, in 2022. He is pursuing his M.S. degree in College of Control Science and Engineering, Zhejiang University, Hangzhou, China.

Chengrui Zhu received the B.Eng. degree in control science and engineering from Zhejiang University, Hangzhou, China, in 2022, where he is currently pursuing the M.S. degree with the Institute of Cyber Systems and Control, Department of Control Science and Engineering, Zhejiang University, Hangzhou, China. His research interests include reinforcement learning and intelligent control.

Jiandang Yang received the B.S. degree in computer science and technology from Shanghai University of Electric Power, Shanghai, China, in 2008, and the M.S. degree in computer application and technology from Hangzhou Dianzi University, Hangzhou, China, in 2013. He is currently an Assistant Research Fellow with the Institute of Cyber-Systems and Control, Zhejiang University, Hangzhou. His current research interests include machine learning, big data mining, and computer vision.

Xiaobin Wei received the B.S degree of Electronic Information Science and Technology Engineering from East China Normal University, Shanghai, China, in 2003.And Master degree of Engineering Science in Telecommunication from The University of New South Wales, Austalia, in 2005. She is currently working in the Radio and television operators of Wasu Broadcast & TV Network CO., Ltd. and has been awarded the title Senior Engineer. Her research interests include networking quality optimization and applications of Artificial Intelligence Technology.

Yunliang Jiang received the Ph.D. degree in computer science and technology from Zhejiang University, Hangzhou, China, in 2006. He is currently a Professor with the School of Computer Science and Technology, Zhejiang Normal University, Jinhua, China, and also with the School of Information Engineering, Huzhou University, Huzhou, China. His research interests include intelligent information processing and geographic information systems.

Yong Liu (Member, IEEE) received his B.S. degree in computer science and engineering from Zhejiang University in 2001, and the Ph.D. degree in computer science from Zhejiang University in 2007. He is currently a professor in the Institute of Cyber Systems and Control, Department of Control Science and Engineering, Zhejiang University. He has published more than 30 research papers in machine learning, computer vision, information fusion, robotics. His latest research interests include machine learning, robotics vision, information processing and granular computing.