



OPEN

A multi-agent reinforcement learning based approach for automatic filter pruning

Zhemin Li, Xiaojing Zuo, Yiping Song, Dong Liang & Zheng Xie[✉]

Deep Convolutional Neural Networks (DCNNs), due to their high computational and memory requirements, face significant challenges in deployment on resource-constrained devices. Network Pruning, an essential model compression technique, contributes to enabling the efficient deployment of DCNNs on such devices. Compared to traditional rule-based pruning methods, Reinforcement Learning(RL)-based automatic pruning often yields more effective pruning strategies through its ability to learn and adapt. However, the current research only set a single agent to explore the optimal pruning rate for all convolutional layers, ignoring the interactions and effects among multiple layers. To address this challenge, this paper proposes an automatic Filter Pruning method with a multi-agent reinforcement learning algorithm QMIX, named QMIX_FP. The multi-layer structure of DCNNs is modeled as a multi-agent system, which considers the varying sensitivity of each convolutional layer to the entire DCNN and the interactions among them. We employ the multi-agent reinforcement learning algorithm QMIX, where individual agent contributes to the system monotonically, to explore the optimal iterative pruning strategy for each convolutional layer. Furthermore, fine-tuning the pruned network using knowledge distillation accelerates model performance improvement. The efficiency of this method is demonstrated on two benchmark DCNNs, including VGG-16 and AlexNet, over CIFAR-10 and CIFAR-100 datasets. Extensive experiments under different scenarios show that QMIX_FP not only reduces the computational and memory requirements of the networks but also maintains their accuracy, making it a significant advancement in the field of model compression and efficient deployment of deep learning models on resource-constrained devices.

Keywords Filter pruning, QMIX algorithm, Knowledge distillation

Deep Convolutional Neural Networks(DCNNs) have achieved groundbreaking advancements in domains such as computer vision, natural language processing, and speech recognition. In computer vision, DCNNs have achieved state-of-the-art performance in image classification, object detection, semantic segmentation, and image generation^{1–6}. Recent research has also extended to medical image encryption, demonstrating the potential in enhancing security and efficiency⁷. In natural language processing, DCNNs combined with recurrent neural networks (RNNs) have significantly improved machine translation and text summarization systems⁸. Recent researches have also shown promise in modeling complex linguistic and cognitive processes^{9,10}. In addition, in speech recognition, the use of DCNNs has led to more accurate acoustic models, contributing to better performance in automatic speech recognition systems¹¹.

However, with the deep learning tasks becoming increasingly complex, DCNNs tend to grow larger in scale, demanding dramatic computational resources^{12,13}. Model compression accelerates the inference process by reducing the model scale and computational complexity, thereby facilitating deployment and real-time inference in practical applications. Research focusing on model compression techniques for DCNNs has become indispensable. It plays a pivotal role in making deep learning feasible on resource-constrained platforms such as mobile devices, embedded systems, or edge computing devices. The model compression techniques primarily conclude network pruning, parameter quantization, low-rank decomposition, and knowledge distillation¹⁴. Network pruning and knowledge distillation are the two most widely used techniques among them.

Network pruning effectively reduces the model parameters and model computation by pruning the units while maintaining the original model performance as much as possible¹⁵. The network pruning process typically involves three steps: (1) Evaluate the importance of units and select the unimportant units for pruning. (2) According to the rule-based prune strategy, iteratively removes relatively unimportant units to reduce the

College of Sciences, National University of Defense Technology, 410073 Changsha, China. [✉]email: xiezheng81@nudt.edu.cn

model scale. (3) Fine-tuning the pruned model to improve model performance. The unit importance evaluation function, pruning strategy, and fine-tuning method are key to the effectiveness of network pruning. This paper focuses on iterative filter pruning of DCNNs, considering that when the pruning unit is the filter in convolutional layers, it can be better matched with hardware accelerators to achieve efficient computations, such as FPGAs and ASICs^{16,17}. In the experiments, we define six explainable feature importance evaluation functions to identify the relatively unimportant filters. By comparing the outcomes of different evaluation functions, the most effective one can be selected for a given scenario or even combined. This approach aims to demonstrate that the proposed method can be more adaptable and generalizable by using multiple evaluation functions.

Traditional rule-based network pruning is limited by the rules set by experts, making it difficult to achieve a globally optimal solution. These methods often rely heavily on domain-specific knowledge and can be highly subjective, limiting their adaptability and generalization across different network architectures. Reinforcement Learning (RL)-based automatic pruning offers a promising alternative by mitigating these issues and reducing subjectivity. RL enables the discovery of optimal pruning policies through interaction with the environment, allowing for more dynamic and adaptive pruning strategies. However, current RL-based research primarily uses a single agent to explore the optimal pruning rate for all convolutional layers, neglecting the complex interactions and effects between multiple layers^{18–20}.

To address these limitations, this study proposes a novel approach that integrates Multi-Agent Reinforcement Learning (MARL) with Deep Convolutional Neural Networks (DCNNs) for more effective pruning. Six filter importance evaluation functions are defined to capture different aspects of filter importance. These functions serve as the basis for our MARL framework, where each agent is responsible for optimizing the pruning rate of a specific layer. By using the MARL algorithm, the agents can collaborate effectively, taking into account the interdependencies between different layers.

The QMIX algorithm²¹ is a well-established benchmark in MARL, known for its ability to solve global optimization problems through Centralized Training with Decentralized Execution (CTDE). QMIX stands for “Q-value Mixing,” which refers to the process of combining the Q-values of individual agents to form a joint Q-value that represents the overall utility or reward of the team’s actions. Motivated by the principles of MARL and the effectiveness of the QMIX algorithm, we propose a DCNN pruning framework based on the QMIX algorithm, as illustrated in Fig. 1. Each convolutional layer is treated as an agent, and the entire DCNN serves as the environment, forming a multi-agent system. Thus, the network pruning task is modeled as a cooperative game. The common goal of all agents is to ensure that the pruned network meets the pruning rate constraint while minimizing the model’s prediction loss as much as possible.

In addition, Knowledge Distillation(KD)²² draws upon the educational concept of transferring knowledge from teacher to student. It transfers the knowledge of a large-scale Teacher Network (T-Net) to a small-scale Student Network (S-Net), where the T-Net is pre-trained with good performance. Therefore, the S-Net can efficiently learn the generalization of the T-Net while remaining small in scale. Motivated by this, we fine-tune the pruned model utilizing KD by setting the original model as T-Net and the pruned model as S-Net.

In conclusion, our primary contribution can be summarized as follows:

- We propose a filter pruning method based on the multi-agent reinforcement learning algorithm QMIX to explore the optimal pruning rate for each agent(layer). It is an automatic network pruning method.

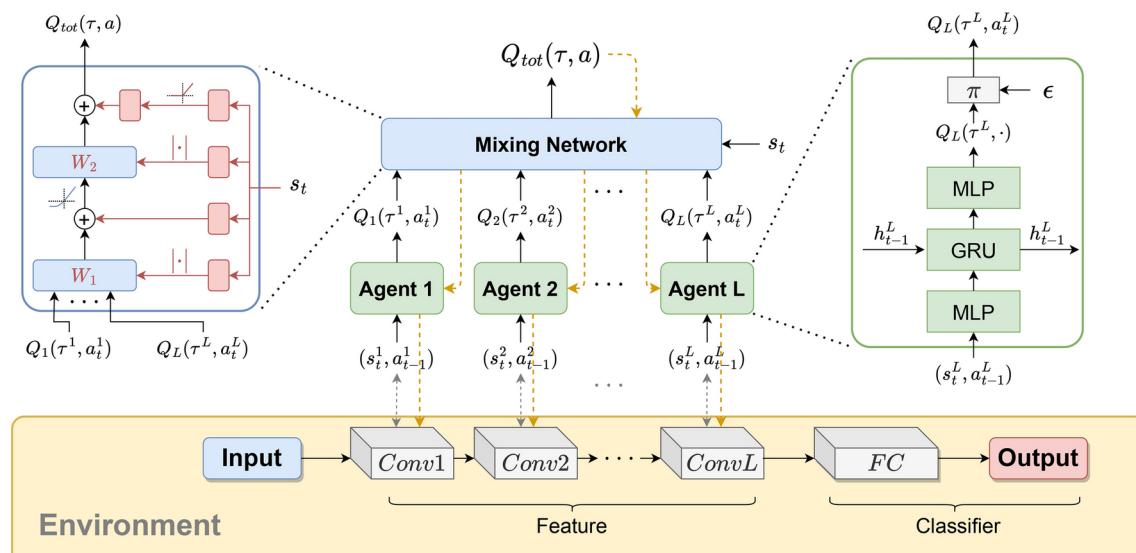


Fig. 1. The environment of DCNN network pruning based on QMIX. Each convolutional layer corresponds to an agent; the whole DCNN is an environment. QMIX consists of L agent networks and a mixing network. The optimal pruning strategy of DCNN is explored using the QMIX algorithm.

- We design an algorithm that integrates network pruning with knowledge distillation, i.e., fine-tuning the pruned model based on KD to accelerate performance improvement.
- We validate the efficiency of the proposed method for DCNN pruning on CIFAR-10 and CIFAR-100 datasets, including VGG-16 and AlexNet. The experimental results show that the QMIX_FP achieves competitive parameter compression and the FLOPs reduction while maintaining accuracy similar to the original network.

The layout of this paper is organized as follows. We overview related works about existing network pruning methods in the Sect. “[Related work](#).” Section “[Methodology](#)” describes our approach to automatic filter pruning based on multi-agent reinforcement learning. We present experimental results and provide a comprehensive analysis of the findings in the Sect. “[Experiments](#),” followed by the Sect. “[Conclusions](#).”

Related work

Network pruning has attracted significant attention in deep learning as an effective model compression technique. It reduces the model scale and computational complexity by removing redundant weights or entire structural components within the neural network while maintaining the performance of the original model to the greatest extent. Research on network pruning has experienced rapid development, including exploring pruning strategies, optimization objectives, automation methods, and so on^{23,24}.

According to the granularity of the pruning unit, network pruning methods can be divided into unstructured pruning (fine-grained) and structured pruning (coarse-grained)¹⁴. Unstructured pruning methods can remove neuron connections (weights) independently. In 2015, Han et al.²⁵ conducted the first real research on deep neural network pruning. It prunes the connections whose absolute values of weight parameters are lower than the threshold and proposes the three-stage iterative pruning process for the first time. This method, however, can lead to highly sparse models that are difficult to deploy on standard hardware. This team further proposed the “Deep Compression” method²⁶. They implemented weight quantization to facilitate weight sharing and applied Huffman coding to achieve better compression. Despite its effectiveness, the method requires additional steps like quantization and coding, which can be complex and time-consuming. Yang et al.²⁷ proposed the energy-aware weight pruning algorithm for CNNs that directly uses energy consumption estimation to guide the pruning process and then locally fine-tuned with a closed-form least-square solution to quickly restore the accuracy. Sanh et al.²⁸ proposed movement pruning, a method that selects weights that tend to move away from zero and applies it to pre-trained language representations (such as BERT). However, these approaches may not generalize well to different hardware platforms.

Conversely, structured pruning focuses on removing entire channels, filters, or layers. Structured pruning methods maintain the regularity of the model structure, making them more hardware-friendly and easier to implement on existing platforms. However, they may not achieve the same level of compression as unstructured pruning. He et al.²⁹ introduced an iterative two-step algorithm to channel prune by a LASSO regression-based channel selection and least square reconstruction. This method can effectively reduce the model size but may sacrifice some accuracy. Li et al.³⁰ pruned filters together with their connecting feature maps from CNNs to reduce the computation costs significantly and can work with existing efficient BLAS libraries for dense matrix multiplications. Liu et al.³¹ made the scaling factors in batch normalization layers towards zero with L_1 regularization to identify insignificant channels and prune. This approach can lead to more compact models but requires careful tuning of hyperparameters. Zhuang et al.³² improved this evaluation method by proposing a polarization regularizer, pushing scaling factors of unimportant neurons to 0 and others to a value $a > 0$. This method enhances the robustness of the pruning process. In summary, structured pruning methods maintain the regularity of the model structure, making them more hardware-friendly and easier to implement on existing platforms, which is why this paper chooses the filter-based structured pruning method.

According to the pruning process, network pruning methods can be categorized into traditional rule-based and automatic RL-based. Rule-based methods typically execute the three-stage iterative pruning process²⁵, where the pruning rate at each step and the total number of iterative are designed in advance^{33,34}. Instead of being confined by subjective rules and hyper-parameter design, some researchers have formulated the network pruning problem as an RL problem to explore optimal solutions automatically. He et al.¹⁸ proposed AutoML for Model Compression (AMC), which leverages reinforcement learning to provide the pruning strategy layer-wise. Gupta et al.²⁰ built upon AMC and contributed an improved framework based on Deep Q-Network(DQN) that provides rewards at every pruning step. Camci et al.³⁵ introduced a deep Q-learning-based pruning method called QLP, which is weight-level unstructured pruning. Feng et al.¹⁹ proposed a filter-level pruning method based on the Deep Deterministic Policy Gradient(DDPG) algorithm. Unfortunately, in existing single-agent RL-based pruning methods, the agent prunes only one layer at each time step, which fails to capture the complex dependencies among multiple layers. This limitation can lead to suboptimal pruning decisions, as the performance of one layer can significantly influence that of subsequent layers. In contrast, multi-agent reinforcement learning (MARL) can better handle these challenges by allowing multiple agents to collaborate and make decisions, thereby improving the overall pruning efficiency and effectiveness.

Existing research has been deeply involved in network pruning for DCNNs, but there are some limitations and untapped potentials in this area. In this paper, we aim to fill the gap in the existing work and propose automatic filter pruning by MARL, which has been under-explored.

Methodology

Our proposed network pruning method, named QMIX_FP, is shown in Fig. 2. This figure provides a comprehensive overview of our approach. Fig. 2b demonstrates explicitly how we utilize the MARL algorithm QMIX for pruning. Here, each agent aligns with a convolutional layer in the DCNN. The network architecture at time step t is represented by the state s_t , while a_t represents the pruning decision made at that time step. The

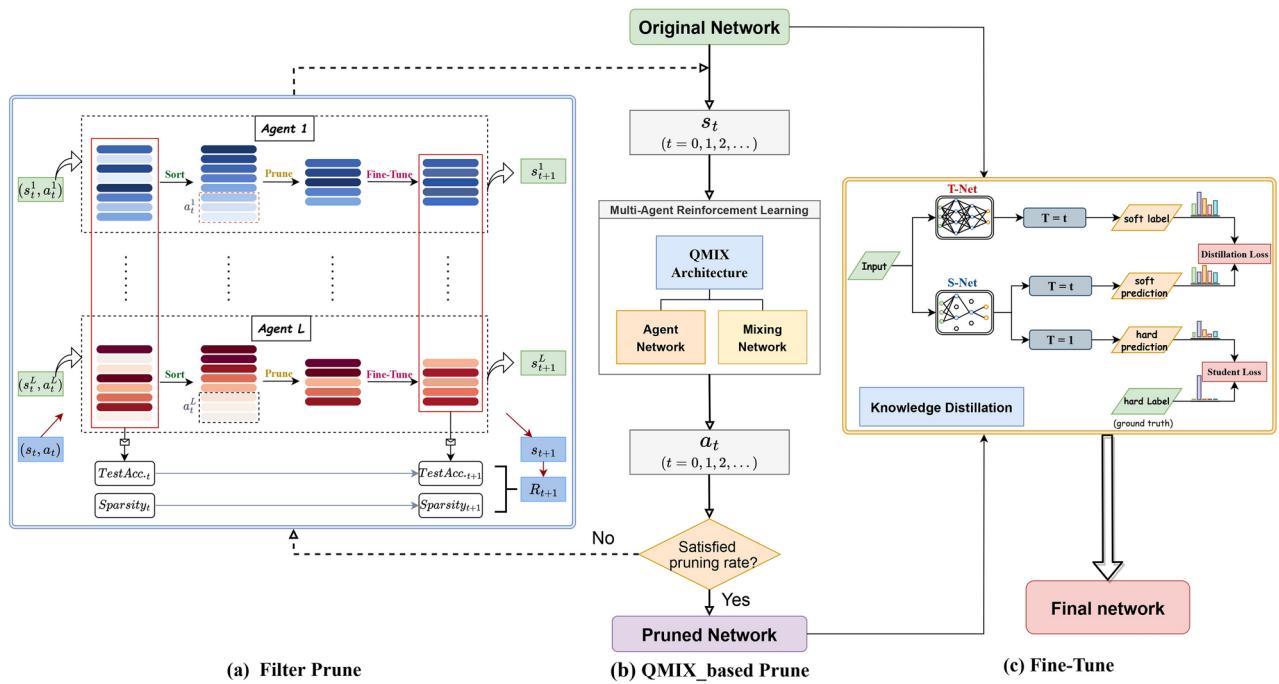


Fig. 2. An overview of our approach QMIX_FP. **(a)** Filters pruning process of Markov Decision Process. **(b)** QMIX-based pruning architecture. **(c)** Knowledge distillation-based fine-tuning structure.

agents progressively test different pruning techniques until the target pruning rate is met, ultimately shaping the pruned network. Figure 2a breaks down the pruning steps at each time step. Once the pruning target is reached, Fig. 2c outlines the distillation phase, where we refine the pruned network and enhance its performance through knowledge distillation. In this section, we initially frame the task of filter-level pruning for DCNNs. We then provide a thorough examination of QMIX_FP, detailing our feature evaluation metrics, the QMIX-driven pruning mechanism, and the KD-based fine-tuning process.

Assume that DCNN contains L convolutional layers. For the i -th convolutional layer L_i , the weight of all filters is denoted as $\mathbf{w}_i \in \mathbb{R}^{n_i \times n_{i-1} \times d_i \times d_i}$, where d_i is the kernel size, n_{i-1} and n_i are input channels and output channels, respectively. $\mathbf{w}_{i,j} \in \mathbb{R}^{n_{i-1} \times d_i \times d_i}$ indicates the j -th filter of L_i . Meanwhile, we denote the bias of L_i is \mathbf{b}_i , and the remaining parameter of DCNN other than the convolutional layer is \mathbf{w}_0 . Thereby, the parameter of DCNN represents as $W = \{\mathbf{w}_0, (\mathbf{w}_1, \mathbf{b}_1), (\mathbf{w}_2, \mathbf{b}_2), \dots, (\mathbf{w}_L, \mathbf{b}_L)\}$. Consider the classification dataset $D = \{X = \{\mathbf{x}_0, \mathbf{x}_1, \dots, \mathbf{x}_{N-1}\}, Y = \{y_0, y_1, \dots, y_{N-1}\}\}$, where y_n is the ground truth of input \mathbf{x}_n and the number of samples is N . For the image classification task, the optimization objective during model training is to minimize the prediction loss function

$$\mathcal{L}(D | W) = \frac{1}{N} \sum_{n=0}^{N-1} \mathbb{I}(\hat{y}_n \neq y_n), \quad (1)$$

where $\mathbb{I}(\cdot)$ indicator function and \hat{y}_n is the predict label of input \mathbf{x}_n .

The filter-level pruning involves removing relatively unimportant filters from the convolutional layers without changing the topology structure and depth of DCNN. After pruning, the parameters of the pruned network are updated to $W' = \{\mathbf{w}'_0, (\mathbf{w}'_1, \mathbf{b}'_1), (\mathbf{w}'_2, \mathbf{b}'_2), \dots, (\mathbf{w}'_L, \mathbf{b}'_L)\}$. To apply mathematical notation for description conveniently, we constrain the W' to be the same dimension as the W . Thus, the optimization objective of the filter pruning task is to minimize the prediction loss of the pruned network when satisfying the overall pruning rate constraint B , i.e.,

$$\min_{W'} \mathcal{L}(D | W) - \mathcal{L}(D | W') \quad s.t. \frac{\|W'\|_0}{\|W\|_0} \leq 1 - B, \quad (2)$$

where $\|\cdot\|_0$ is l_0 norm. Intuitively, the proportion of filters retained in W' should be less than or equal to $1 - B$. Here, B represents the overall pruning rate constraint for the entire DCNN.

Feature evaluation

The feature evaluation function for filter importance computing directly influences the effectiveness of DCNN network pruning with the given pruning rate. Filters play an important role in image understanding and feature extraction in DCNN. The input image data undergoes convolution with filters in the convolutional layer,

resulting in feature maps, where each filter corresponds to a feature map. Typically, an activation function is applied to these feature maps to introduce nonlinearity and increase the expressiveness of the model, resulting in activation maps. In this work, we choose the classic ReLU (Rectified Linear Unit) as the activation function.

Therefore, the importance of filters can be evaluated based on their inherent properties or the information representations generated by the filters, such as feature maps and activation maps. We denote the feature map produced by filter \mathbf{w}_i as $F_i = \{\mathbf{F}_{i,1}, \mathbf{F}_{i,2}, \dots, \mathbf{F}_{i,n_i}\}$ and the activation map produced by F_i as $A_i = \{A_{i,1}, A_{i,2}, \dots, A_{i,n_i}\}$. Considering the explainability of network pruning, we define six feature importance evaluation functions based on feature attribution theory in explainable artificial intelligence³⁶ to demonstrate the adaptability and generalizability of our proposed method. According to the objects used for calculating the importance of filter $\mathbf{w}_{i,j}$, we classify the explainable evaluation functions $\text{Imp}(\cdot)$ into three categories: filter-based evaluation function $\text{Imp}(\mathbf{w}_{i,j})$, feature map-based evaluation function $\text{Imp}(\mathbf{F}_{i,j})$, and activation map-based evaluation function $\text{Imp}(A_{i,j})$. The larger of $\text{Imp}(\mathbf{w}_{i,j})$ implies that the filter plays a more significant role in the reference process. Likewise, the larger $\text{Imp}(\mathbf{F}_{i,j})$ or $\text{Imp}(A_{i,j})$ is, the more important the corresponding filter $\mathbf{w}_{i,j}$ is.

Filter-based

- l_2 Norm of Filter Weight

It's most intuitive to directly evaluate the importance based on the filter's weight parameters. We define that

$$\text{Imp}_{\text{Weight}}(\mathbf{w}_{i,j}) = \frac{1}{|\mathbf{w}_{i,j}|} \sum_{m=1}^{|\mathbf{w}_{i,j}|} w_m^2, \quad (3)$$

where $|\mathbf{w}_{i,j}|$ denotes the modulus of set obtained by $\mathbf{w}_{i,j}$ vectorization.

- Taylor Expansion of Loss Function

The extent of a filter's contribution is indirectly determined by calculating the change of loss function $\mathcal{L}(D | W)$ before and after removing this filter from DCNN. A large change in the loss function signifies that this filter is of high importance. To reduce the computational complexity, we utilize the first-order Taylor expansion of the loss function to approximate it. Thus, we get the approximation importance of each filter only by computing the gradient of $\mathcal{L}(D | W)$ once. The criterion for calculation is

$$\begin{aligned} \text{Imp}_{\text{Taylor}}(\mathbf{w}_{i,j}) &= |\mathcal{L}(D|\mathbf{w}_{i,j}) - \mathcal{L}(D|\mathbf{w}_{i,j} = \mathbf{0})| \\ &= \left| \mathcal{L}(D|\mathbf{w}_{i,j} = \mathbf{0}) + \frac{\partial \mathcal{L}(D|\mathbf{w}_{i,j} = \mathbf{0})}{\partial \mathbf{w}_{i,j}} \mathbf{w}_{i,j} + R_1(\mathbf{w}_{i,j}) - \mathcal{L}(D|\mathbf{w}_{i,j} = \mathbf{0}) \right| \\ &\approx \left| \frac{\partial \mathcal{L}(D|\mathbf{w}_{i,j} = \mathbf{0})}{\partial \mathbf{w}_{i,j}} \mathbf{w}_{i,j} \right|, \end{aligned} \quad (4)$$

where $\mathbf{w}_{i,j} = \mathbf{0}$ indicates that the filter $\mathbf{w}_{i,j}$ is removed, $R_1(\mathbf{w}_{i,j})$ is the remainder term, negligible.

Feature map-based

- Information Bottleneck Value

Motivated by the Information Bottleneck(IB) principle³⁷, we calculate the information bottleneck value of the feature map $\mathbf{F}_{i,j}$, i.e.,

$$\text{Imp}_{\text{IB}}(\mathbf{F}_{i,j}) = \mathbb{I}(\mathbf{F}_{i,j}; Y) - \beta \mathbb{I}(\mathbf{F}_{i,j}; X), \quad (5)$$

where $\mathbb{I}(\cdot ; \cdot)$ is mutual information, and $\beta \in [0, 1]$ is a balancing factor for the trade-off between information compression and information retention. For simplicity, we use Information Gain (IG), derived from information entropy $\mathbb{H}(\cdot)$, to estimate mutual information, i.e.,

$$\begin{cases} \mathbb{I}(\mathbf{F}_{i,j}; Y) = \text{IG}(Y, \mathbf{F}_{i,j}) = \mathbb{H}(Y) - \mathbb{H}(Y | \mathbf{F}_{i,j}) \\ \mathbb{I}(\mathbf{F}_{i,j}; X) = \text{IG}(\mathbf{F}_{i,j}, X) = \mathbb{H}(\mathbf{F}_{i,j}) - \mathbb{H}(\mathbf{F}_{i,j} | X). \end{cases}$$

- γ of Batch Normalization

Batch Normalization layers are typically inserted immediately after the convolution layer but before the activation layer. It ensures that the inputs to the activation functions have a more stable distribution. Assume z_{in} and z_{out} are the input and output of the Batch Normalization(BN) layer. The BN layer performs the following transformation:

$$z_{\text{out}} = \frac{z_{\text{in}} - \mathbb{E}[z_{\text{in}}]}{\sqrt{\text{Var}[z_{\text{in}}] + \epsilon}} \cdot \gamma + \beta. \quad (6)$$

where $\mathbb{E}[z_{\text{in}}]$ and $\text{Var}[z_{\text{in}}]$ are the mean and the variance of the input, and ϵ is a small constant to avoid division by zero. We refer to the channel importance evaluation criterion proposed by Liu et al.³¹ and define the importance of $\mathbf{F}_{i,j}$ as

$$\text{Imp}_{\text{BN}}(\mathbf{F}_{i,j}) = \gamma. \quad (7)$$

If γ is close to 0, it means that $\mathbf{F}_{i,j}$ has significantly been reduced in scale. On the contrary, the large γ means $\mathbf{F}_{i,j}$ is enlarged in scale, which indicates that the influence on feature learning and information transfer is more significant.

- Gradient

The gradient of the loss function with respect to the feature map reflects the extent to which small perturbations in $\mathbf{F}_{i,j}$ affect prediction loss. We define that

$$\text{Imp}_{\text{Gradient}}(\mathbf{F}_{i,j}) = \left| \frac{\partial \mathcal{L}(D | W)}{\partial \mathbf{F}_{i,j}} \right|. \quad (8)$$

Activation map-based

- Average Percentage of Positive

This study refers to the Average Percentage of Zeros (APoZ) criterion proposed by Hu et al.³⁸, and infers that the average percentage of positive value in the activation map $\mathbf{A}_{i,j}$ can evaluate the corresponding filter $\mathbf{w}_{i,j}$'s importance, i.e.,

$$\text{Imp}_{\text{APoP}}(\mathbf{A}_{i,j}) = 1 - \frac{\sum_n \sum_m \mathbb{I}(a_{n,m} = 0)}{N \times M}, \quad (9)$$

where $\mathbb{I}(\cdot)$ is indicator function. The larger $\text{Imp}_{\text{APoP}}(\mathbf{A}_{i,j})$ means the more important filter $\mathbf{w}_{i,j}$.

Network pruning

As shown in Fig. 1, we explore the DCNN pruning strategy based on multi-agent reinforcement learning QMIX. The interaction process between agents and environment is described by a Markov Decision Process(MDP), i.e., a tuple $(\mathcal{S}, \mathcal{A}, P, R, \gamma)$, which indicates the state space, the action space, the rewards, the state-transition probability matrix, and the penalization factors, respectively. Here, it illustrates the first four elements within the tuple in detail.

- State Representation

At time step t , the global state of the environment notes $s_t = \{s_t^1, s_t^2, \dots, s_t^L\}$, where s_t^i is the state of agent i , which is also the state of the i -th convolutional layer. We characterize the state s_t^i with six features:

$$s_t^i = (n_i, n_{i-1}, \text{Kernel_size}_i, \text{Mean_weight}_i, \text{FLOPs}_i, \text{Params}_i), \quad (10)$$

where n_i is the number of filters in the i -th layer, which is also the input channel of the $(i+1)$ -th layer. n_{i-1} is the input channel of the i -th layer, which is also the number of filters in the $(i-1)$ -th layer. Kernel_size_i is the kernel size and Mean_weight_i is the mean of weight. FLOPs_i and Params_i denote the Floating-point Operations(FLOPs) and the Parameters(Params) of the i -th convolutional layer, respectively.

- Action Space

In our research, each agent corresponds to a single convolutional layer within the neural network, with its action defined as the pruning rate applied to that layer, represented by a simple numerical value. To facilitate precise control over the pruning process, we establish the action space as a discrete set of possible pruning rates, where $\mathcal{A} = [0\%, 5\%, 10\%, 15\%, 20\%]$. At time step t , the collective action across all layers is represented as $a_t = \{a_t^1, a_t^2, \dots, a_t^L\}$, where a_t^i indicates the pruning rate selected for the i -th convolutional layer from the predefined set.

These pruning rates are not static and can evolve throughout the reinforcement learning process. The ultimate goal is to iteratively fine-tune these rates to achieve a targeted overall pruning level B for the entire deep convolutional neural network (DCNN). This tuning is done with the dual goals of minimizing the network's prediction error and optimizing its performance, ensuring that the pruned model remains both efficient and accurate.

- State Transition

At time step t , agent i receives the current state s_t^i of the i -th convolutional layer. Firstly, the importance scores of all filters are evaluated and sorted. Then, agent i prunes the filters with lower importance according to the current action a_t^i , which is output by agent network i . Notably, all agents perform this pruning process simultaneously. Finally, we fine-tune the pruned network and receive the global state of the environment s_{t+1} . The above process describes a state transition in the environment, as shown in the upper part of Fig. 2a, which is repeated until the end of the episode.

- Reward Function

Instant reward directly induces DCNN pruning toward optimality. Network pruning aims to minimize model performance drop while achieving the expected pruning rate. Therefore, we define instant reward as a linear combination of test accuracy change and network sparsity change, where the sparsity is the ratio of the total filters in the pruned network to the original network.

$$R_t = \lambda * (\text{TestAcc}_t - \text{TestAcc}_{t-1}) + \mu * (\text{Sparsity}_t - \text{Sparsity}_{t-1}), \quad (11)$$

where λ and μ are hyperparameters, TestAcc_t and Sparsity_t are prediction accuracy on the test dataset and network sparsity at time step t , respectively. In addition, the total return of an episode is the cumulative of instant rewards.

- Loss Function

The MDP provides a mathematical framework for DCNN pruning. We solve the above MDP problem based on the QMIX algorithm, as shown in Fig. 2b. The QMIX architecture comprises L agent networks alongside a mixing network. Each agent i is equipped with its own agent network, which represents the state-action value function $Q_i(\tau^i, a^i)$, where τ signifies the joint action-observation history. The mixing network, a feed-forward neural network, takes the outputs from the agent networks as inputs and combines them in a monotonic manner to generate the total value Q_{tot} . Specifically, Q_{tot} is calculated as follows:

$$Q_{\text{tot}}(\tau, a, s) = \mathcal{M}(Q_1(\tau^1, a^1), Q_2(\tau^2, a^2), \dots, Q_L(\tau^L, a^L)),$$

where \mathcal{M} is the mixing function implemented by the mixing network. This function ensures that the combination of individual agent Q-values is monotonic and adheres to the problem's constraints. s represents the global state, which is necessary for centralized training.

To train the QMIX model, we implement a Temporal-Difference loss function aimed at minimizing the discrepancy between the predicted Q-values and the target Q-values. The training is conducted in an end-to-end fashion to minimize the following loss function:

$$\mathcal{L}(\theta) = \sum_{i=1}^b \left[r + \gamma \max_{a'} Q_{\text{tot}}(\tau', a', s'; \theta^-) - Q_{\text{tot}}(\tau, a, s; \theta) \right]^2, \quad (12)$$

where b is the batch size of transitions sampled from buffer, $(\cdot)'$ indicates the value of (\cdot) at the next time step, θ^- are the parameter of target network.

Fine-tuning

The QMIX algorithm follows the CTDE training principle. After training it to convergence, each agent network makes decisions independently based on its own observations. Given the expected pruning rate, we prune the original pre-trained network according to the pruning strategy output from the trained QMIX's agent networks. The pruned network undergoes a certain level of performance decline, reflecting degradation as a result of the pruning process.

To accelerate the performance improvement of the pruned network, we design a model compression framework that integrates network pruning and knowledge distillation, as shown in Fig. 2c. Specifically, we set the original pre-trained network as T-Net and the pruned network as S-Net. Then fine-tuning the pruned network based on knowledge distillation.

On the one hand, to guide S-Net to more fully learn the generalization ability of T-Net, we introduce label probability distributions with a temperature parameter T ,

$$p_i(z_i, T) = \frac{\exp(z_i/T)}{\sum_{j=0}^{k-1} \exp(z_j/T)}, \quad (13)$$

Compute the label probability distribution of S-Net and T-Net for $T = t$ ($t > 1$), respectively. We define the cross-entropy loss between two distributions as the distillation loss,

$$\mathcal{L}_D(p(z_t, T), p(z_s, T)) = - \sum_{i=0}^{k-1} p_i(z_{ti}, T=t) \log(p_i(z_{si}, T=t)), \quad (14)$$

where z_t and z_s are the fully connected layer outputs of T-Net and S-Net, respectively, and p_i denotes the probability of the i -th label. On the other hand, S-Net needs to learn not only the knowledge from T-Net but also the ground truth. We define the cross-entropy loss between label probability distributions for $T = 1$ and the ground truth as the student loss,

$$\mathcal{L}_S(y, p(z_s, T)) = - \sum_{i=0}^{k-1} y_i \log(p_i(z_{si}, T=1)). \quad (15)$$

With the above discussion, the optimization objective of fine-tuning the pruned network based on KD is a linear combination of the distillation loss and the student loss,

$$\mathcal{L} = \alpha * \mathcal{L}_D(p(z_t, T), p(z_s, T)) + (1 - \alpha) * \mathcal{L}_S(y, p(z_s, T)), \quad (16)$$

where α is a hyperparameter that controls the weights of the two losses.

In summary, Algorithm 1 describes the process of automatic filter pruning based on QMIX.

Input: Pre-trained DCNN; Dateset D ; Pruning rate B ; Batch size M ; Imp(\cdot); Prune(\cdot).

Output: Pruned_DCNN.

- 1: **Stage 1: Train QMIX.**
- 2: Initialize QMIX, episode $\leftarrow 0$.
- 3: **while** episode $<$ max_episodes **do**
- 4: DCNN₀ \leftarrow DCNN.
- 5: $t \leftarrow 0$.
- 6: Initialize environment, Terminated _{t} = False, recieve s_0^i and a_0^i , for $i = 1, 2, \dots, L$.
- 7: **while** $t <$ max_step or Terminated _{t} = True **do**
- 8: Obtaining action of agent i : $a_t^i, Q_t^i = Q_i(s_t^i, a_{t-1}^i)$, for $i = 1, 2, \dots, L$.
- 9: Select filters to be prune: Filters_{de} = Imp(DCNN _{t} , a_t).
- 10: Prune DCNN: DCNN _{$t+1, s_{t+1}$} = Prune(DCNN _{t} , Filters_{de}).
- 11: Update TestAcc _{t} and Sparsity _{t} , get reward R_t according to equation (11).
- 12: **if** Sparsity _{t} $\geq B$ **then**
- 13: Terminated _{t} = True.
- 14: **end if**
- 15: Record transition = ($s_t, a_t, s_{t+1}, R_t, \text{Terminated}_t$) into Buffer.
- 16: **if** len(Buffer) $\geq M$ **then**
- 17: Sample batch transitions from Buffer and train QMIX according to equation (12).
- 18: **end if**
- 19: $t \leftarrow t + 1$
- 20: **end while**
- 21: episode \leftarrow episode + 1
- 22: **end while**
- 23: **return** Trained QMIX.

- 24: **Stage 2: Network Pruning.**
- 25: Get Pruned_DCNN through agent network of trained QMIX.
- 26: T_Net \leftarrow DCNN, S_Net \leftarrow Pruned_DCNN.
- 27: Fine tuning S_Net according to equation (16).
- 28: Pruned_DCNN \leftarrow S_Net.
- 29: **return** Pruned_DCNN.

Algorithm 1. QMIX_FP: Automatic DCNN filter pruning based on QMIX.

Experiments

Experimental settings

To demonstrate the efficiency of our proposed method QMIX_FP, we experiment on the classical deep convolutional neural networks VGG-16³⁹ and AlexNet1. VGG-16 consists of 13 convolutional layers and 3 fully connected layers. The filters are distributed as 64, 64, 128, 128, 256, 256, 256, 512, 512, 512, 512, 512, 512. AlexNet consists of 5 convolutional layers and 3 fully connected layers. The filter distribution is 64, 192, 384, 256, 256. We test their performance over CIFAR-10 and CIFAR-100 datasets⁴⁰ to further evaluate network pruning effectiveness.

In our experimental setting, we adopt widely used metrics to evaluate the network pruning effectiveness, including prediction accuracy, the number of filters, Parameters(Params), and Floating-point Operations(FLOPs). Notably, we set the prediction accuracy to be task-specific, i.e., top-1 accuracy on CIFAR-10 and top-5 accuracy on CIFAR-100.

For the original pre-trained DCNNs, we set the batch size as 128 and updated the network with the stochastic gradient descent algorithm, training for 200 epochs. The initial learning rate is 0.1, the decay rate is 1e-4, and the momentum is 0.9. For network pruning based on QMIX, we set that the discounted return is 0.99, and the λ and μ in the reward function are 0.7 and 0.3, respectively. Moreover, the QMIX training optimizer is Adam, the learning rate is 0.01, the Buffer size is 512, the max episode is 500, and the max time step is 20 in an episode. The target network updates its parameters every 5 time steps. For network fine-tuning, we set the T in distillation loss as 2.5, the coefficient of distillation loss α as 0.3, and training epochs as 100.

Results and analysis

We first conducted experimental analysis on the pruning result of VGG-16 on the benchmark dataset CIFAR-10. To further verify the robustness of our approach QMIX_FP, we successively replaced the DCNN with AlexNet and replaced the dataset with the more complex CIFAR-100, performing network pruning and analyzing results.

VGG-16 + CIFAR-10

To thoroughly investigate the performance and stability of the QMIX_FP under different target pruning rates, we set three expected pruning rates of 50%, 60%, and 70%, to observe the network compression results with various sparsity levels. Under the constraint of pruning rate, we prune VGG-16 with six filter importance evaluation functions, respectively. The results are shown in Table 1. The “Base” row is the metrics of the original pre-trained VGG-16. The “Acc. (pruned)” column shows the prediction accuracy of the pruned network. The “Acc. (fine-

		Pruning rate: 50%				
Imp(·)		Acc.(pruned)	Acc.(fine-tuned)	Filters	Params(M)	FLOPs(M)
	Base	0.9356	–	4224	14.918602	314.501632
$w_{i,j}$	Weight	0.9141	0.9228[98.63%]	2030	3.183586	95.409176
	Taylor	0.9141	0.9243[98.79%]	2081	4.008244	144.358784
$F_{i,j}$	IB	0.9141	0.9319[99.60%]	2034	3.130686	175.470180
	BN	0.9219	0.9274[99.12%]	2092	3.156463	132.893916
	Gradient	0.9141	0.9204[98.38%]	2057	2.452875	93.555904
$A_{i,j}$	APoP	0.8828	0.9230[98.65%]	2018	3.015264	113.696572
		Pruning rate: 60%				
Imp(·)		Acc.(pruned)	Acc.(fine-tuned)	Filters	Params(M)	FLOPs(M)
	Base	0.9356	–	4224	14.918602	314.501632
$w_{i,j}$	Weight	0.9219	0.9304[99.44%]	1665	2.160138	138.856536
	Taylor	0.8750	0.9209[98.43%]	1637	1.798093	111.056384
$F_{i,j}$	IB	0.8750	0.9245[98.81%]	1681	2.748931	152.135916
	BN	0.9209	0.9221[98.56%]	1602	2.080667	92.665820
	Gradient	0.8984	0.9279[99.18%]	1675	2.002832	160.207432
$A_{i,j}$	APoP	0.8438	0.9165[97.96%]	1679	2.119400	90.689736
		Pruning rate: 70%				
Imp(·)		Acc.(pruned)	Acc.(fine-tuned)	Filters	Params(M)	FLOPs(M)
	Base	0.9356	–	4224	14.918602	314.501632
$w_{i,j}$	Weight	0.8281	0.8929[<u>95.44%</u>]	1260	0.693829	47.611672
	Taylor	0.8047	0.8975[95.93%]	1253	0.905252	56.988912
$F_{i,j}$	IB	0.8438	0.9190[98.23%]	1266	1.721641	120.906412
	BN	0.8575	0.9238[98.74%]	1257	1.270740	111.091596
	Gradient	0.8750	0.9256[98.93%]	1258	1.715084	140.446552
$A_{i,j}$	APoP	0.6563	0.8992[96.11%]	1183	1.054153	52.143756

Table 1. Pruning results of VGG-16 on CIFAR-10. The best pruning results are bolded, and the worst results are underlined.

tuned)" column shows the fine-tuned prediction accuracy based on the knowledge distillation, and the data in square brackets shows the percentage of the prediction accuracy compared to the base. The "Filters" column indicates the number of filters. "M" denotes millions in the columns of FLOPs and Params columns.

Analyzing the prediction accuracy of the pruned network after fine-tuning, it's shown that when the pruning rate is 50%, the pruned network maintains 99.60% of the original VGG-16 based on evaluation function ImpIB($\mathbf{w}_{i,j}$), reaching the highest level. At a pruning rate of 60%, the pruned network maintains 98.73% of the original VGG-16, based on the average of the six importance evaluation methods. When the pruning rate climbs to 70%, the performance is 97.23% of the original. These results fully demonstrate the expected effect of network pruning, substantiating the efficacy of our proposed pruning method based on the QMIX algorithm and fine-tuning method based on knowledge distillation. Comparing six filter evaluation methods, we find that the pruned network obtained by the QMIX algorithm performs better only when evaluated based on the γ factor of the batch normalization layer. In contrast, it performs slightly less effectively when evaluated based on the APoP of the activation maps. What's more, with the given pruning rate, the FLOPs and Params of the pruned network tend to be compressed by a larger amount. For example, when the pruning rate is 50%, the Params is compressed by about 77.50% based on evaluation function ImpWeight($\mathbf{w}_{i,j}$); when the pruning rate is 60%, the FLOPs is compressed by about 76.65% based on evaluation function ImpTaylor($\mathbf{w}_{i,j}$).

AlexNet + CIFAR-10

To verify that the network pruning applies to different DCNNs, we prune the pre-trained AlexNet and evaluate its performance on the benchmark dataset CIFAR-10. AlexNet's architecture consists of five convolutional layers with a total of 1152 filters, resulting in fewer parameters and computations compared to VGG-16. We get a prediction accuracy of 0.8585 for the original pre-trained AlexNet on CIFAR-10, which is due to its lower complexity than VGG-16. The pruning result of AlexNet according to QMIX_FP is shown in Table 2, including indicators of changes in prediction accuracy and network structure parameters.

We observe that when the pruning rate of AlexNet is at 50% or 60%, the pruned networks' prediction accuracy can maintain approximately 99.00% of the original accuracy. At the pruning rate of 70%, the network's performance is slightly degraded, but it still manages to maintain over 95.49% of its original accuracy. In comparison, when the feature evaluation function depends on the information bottleneck value of the filter's feature map, the pruned network suffers a greater loss of accuracy. Further, when the network pruning rates are 50%, 60%, and 70%, the pruned network's Params are approximately 31.19%, 23.22%, and 15.46% of the original network, respectively. The AlexNet achieves a higher level of compression in both space complexity and

		Pruning rate: 50%				
Imp(·)		Acc.(pruned)	Acc.(fine-tuned)	Filters	Params(M)	FLOPs(M)
	Base	0.8585	–	1152	2.714826	39.823296
$\mathbf{w}_{i,j}$	Weight	0.8516	0.8480[98.78%]	574	0.871555	14.119357
	Taylor	0.8203	0.8373[97.53%]	551	0.754758	14.956772
$\mathbf{F}_{i,j}$	IB	0.7734	0.8413[98.00%]	570	0.880309	18.481240
	BN	0.8438	0.8470[98.66%]	560	0.827679	18.891221
	Gradient	0.8438	0.8520[99.24%]	575	0.885663	21.158055
$\mathbf{A}_{i,j}$	APoP	0.8516	0.8489[98.88%]	550	0.860180	20.911694
		pruning rate: 60%				
Imp(·)		Acc.(pruned)	Acc.(fine-tuned)	Filters	Params(M)	FLOPs(M)
	Base	0.8585	–	1152	2.714826	39.823296
$\mathbf{w}_{i,j}$	Weight	0.8906	0.8472[98.68%]	459	0.596856	15.469136
	Taylor	0.8047	0.8349[97.25%]	446	0.647585	16.149933
$\mathbf{F}_{i,j}$	IB	0.7969	0.8346[97.21%]	454	0.615360	15.795638
	BN	0.7969	0.8452[98.45%]	456	0.631417	16.397015
	Gradient	0.7969	0.8353[97.30%]	460	0.633874	13.138188
$\mathbf{A}_{i,j}$	APoP	0.7656	0.8423[98.11%]	455	0.658449	16.962633
		pruning rate: 70%				
Imp(·)		Acc.(pruned)	Acc.(fine-tuned)	Filters	Params(M)	FLOPs(M)
	Base	0.8585	–	1152	2.714826	39.823296
$\mathbf{w}_{i,j}$	Weight	0.8047	0.8321[96.92%]	345	0.414934	7.052608
	Taylor	0.7109	0.8171[95.18%]	337	0.393662	6.878678
$\mathbf{F}_{i,j}$	IB	0.6250	0.7799[90.84%]	328	0.380263	7.279463
	BN	0.7891	0.8326[96.98%]	340	0.449226	12.486408
	Gradient	0.7812	0.8366[97.45%]	336	0.471698	15.172924
$\mathbf{A}_{i,j}$	APoP	0.6328	0.8204[95.56%]	334	0.408761	9.024843

Table 2. Pruning results of AlexNet on CIFAR-10. The best pruning results are bolded, and the worst results are underlined.

time complexity, which satisfies the requirements in practical application scenarios. Thus, it can be seen that the QMIX_FP algorithm is not only applicable to high-complexity networks but also effective to small-scale DCNNs, which demonstrates its broad applicability in neural network architectures of different sizes.

VGG-16 + CIFAR-100

One metric for evaluating the effectiveness of network pruning is the prediction accuracy achieved on image classification datasets; hence, the dataset is also an influencing factor. We still perform network pruning on VGG-16 but evaluate its performance on the more complex CIFAR-100 dataset, where the prediction accuracy is measured using the top-5 accuracy. As shown in Table 3, we find the pruned network obtained by ImpIB($\mathbf{F}_{i,j}$) is generally worse compared to other evaluation functions. Fine-tuning the pruned network through knowledge distillation can improve the prediction accuracy to 95.51% - 99.05% of the original pre-trained model. However, the model exhibits significant performance degradation under higher pruning rate constraints. The experimental results show that our algorithm QMIX_FP is still effective on the complex dataset CIFAR-100.

To intuitively judge the rationality of exploring the optimal pruning strategy via the multi-agent reinforcement learning algorithm QMIX, we visualize the changes in the number of filters within each convolutional layer of VGG-16 and AlexNet both before and after pruning, as shown in Fig. 3. Figure a shows the filter number distribution of VGG-16 when the pruning rate is 50% where the evaluation function is ImpIB($\mathbf{F}_{i,j}$). For the pruned network, the number of filters in each convolutional layer is 58, 64, 92, 128, 256, 256, 181, 109, 166, 247, 195, 116, 166. Figure b shows the filter number distribution of AlexNet when the pruning rate is 50% where the evaluation function is ImpGradient($\mathbf{F}_{i,j}$). After pruning, the number of filters in each convolutional layer is 56, 183, 133, 114, and 89. We find that the convolutional layers closer to the classifier have a higher proportion of redundant filters, which is also associated with the larger number of filters contained.

To demonstrate the advantages of our proposed method QMIX_FP, we compare our pruning results with the HRank⁴¹, HBFP⁴² and DDPG_F¹⁹. Additionally, we compare the method employing a randomized evaluation criterion, i.e., the importance score of each filter is a random number. We adopt the same experimental setup as the baseline methods and focus on the pruning results of VGG-16 only.

- HRank is an iterative filter pruning approach. The main idea is that low-rank feature maps contain less information, and the corresponding filters are considered relatively unimportant and can be pruned.
- HBFP utilizes network training history for filter pruning iteratively. It prunes one redundant filter from each filter pair, ensuring minimal information loss during network training.

		pruning rate: 50%				
Imp(·)		Acc.(pruned)	Acc.(fine-tuned)	Filters	Params(M)	FLOPs(M)
	Base	0.8961	–	4224	14.941732	314.524672
$\mathbf{w}_{i,j}$	Weight	0.8672	0.8856[98.82%]	2020	3.043907	135.09748
	Taylor	0.8281	0.8755[97.70%]	2026	3.073754	100.467832
$\mathbf{F}_{i,j}$	IB	0.7969	0.8737[97.50%]	2097	3.313371	137.484096
	BN	0.9062	0.8876[99.05%]	2099	3.008191	167.201600
	Gradient	0.8047	0.8809[98.30%]	2095	3.264122	136.683472
$\mathbf{A}_{i,j}$	APoP	0.7969	0.8786[98.04%]	2025	3.187321	160.232956
		pruning rate: 60%				
Imp(·)		Acc.(pruned)	Acc.(fine-tuned)	Filters	Params(M)	FLOPs(M)
	Base	0.8961	–	4224	14.941732	314.524672
$\mathbf{w}_{i,j}$	Weight	0.7969	0.8773[97.90%]	1611	2.117810	72.620072
	Taylor	0.8594	0.8790[98.09%]	1675	2.131325	121.722136
$\mathbf{F}_{i,j}$	IB	0.6250	0.8665[96.69%]	1673	2.582921	98.543556
	BN	0.8203	0.8759[97.74%]	1619	1.911082	71.725748
	Gradient	0.7578	0.8718[97.28%]	1660	2.069356	84.059268
$\mathbf{A}_{i,j}$	APoP	0.6016	0.8681[98.11%]	1647	2.051498	96.258680
		pruning rate: 70%				
Imp(·)		Acc.(pruned)	Acc.(fine-tuned)	Filters	Params(M)	FLOPs(M)
	Base	0.8961	–	4224	14.941732	314.524672
$\mathbf{w}_{i,j}$	Weight	0.6406	0.8657[96.60%]	1220	0.889273	51.890312
	Taylor	0.7266	0.8727[97.39%]	1193	1.075607	72.818532
$\mathbf{F}_{i,j}$	IB	0.4688	0.8626[96.26%]	1259	1.052788	88.585496
	BN	0.7656	0.8707[97.16%]	1238	1.286112	50.040904
	Gradient	0.7109	0.8730[97.42%]	1220	1.267270	92.931200
$\mathbf{A}_{i,j}$	APoP	0.3828	0.8559[95.51%]	1246	1.282633	45.668076

Table 3. Pruning results of VGG-16 on CIFAR-100. The best pruning results are bolded, and the worst results are underlined.

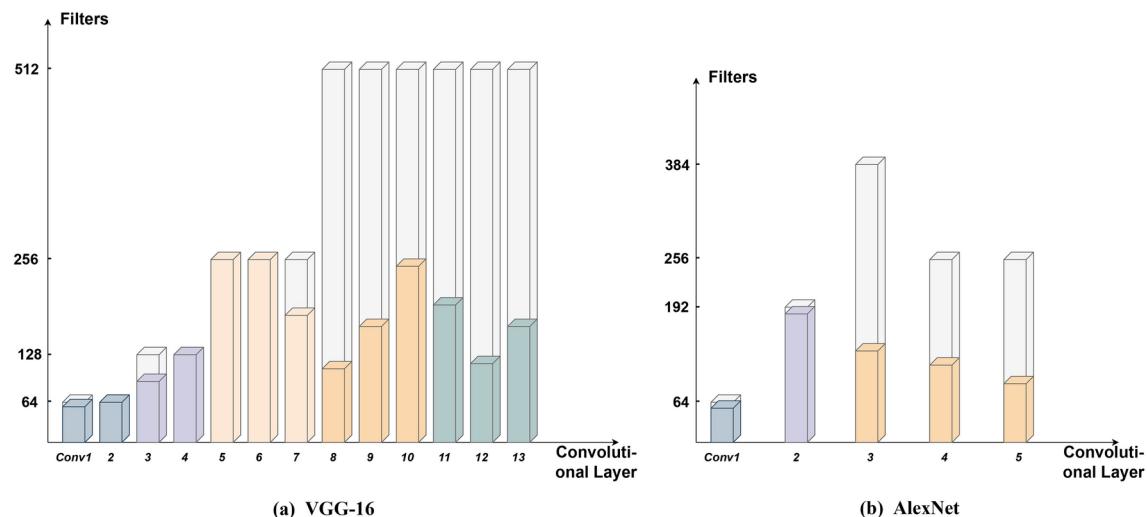


Fig. 3. The changes in the number of filters within each convolutional layer of VGG-16 and AlexNet. In the bar chart, the gray bars represent the original number of filters in each layer, while the colored bars represent the number of filters in each layer of the pruned network.

Datasets	Methods	Acc. (base)	Acc. (new)	Acc. \downarrow (%)
CIFAR-10	Random	0.9356	0.8882	5.07
	HRank	0.9396	0.9234	1.72
	HBFP	0.9396	0.9199	2.10
	DDPG_FP	0.9367	0.9324	0.46
	QMIX_FP	0.9356	0.9319	0.40
CIFAR-100	Random	0.8961	0.8047	10.20
	DDPG_FP	0.7171	0.6912	3.61
	QMIX_FP	0.8961	0.8876	0.95

Table 4. The comparative pruning results for VGG-16, where the Acc. \downarrow indicates the percentage decrease in prediction accuracy after pruning. The best pruning results are bolded.

- DDPG_FP is a filter-level pruning method based on Deep Deterministic Policy Gradient (DDPG), a single-agent reinforcement learning algorithm. The agent explores the pruning rate of one convolutional layer for each time step, and the pruned network is fine-tuned by retraining.

HRank and HBEP are traditional rule-based pruning methods, while DDPG_FP and our method QMIX_FP are automatic RL-based pruning methods. As shown in Table 4, the pruned network from RL-based methods exhibits lower model degradation than traditional methods. This is because the reinforcement learning algorithm can explore a larger pruning strategy space and continuously update the strategy according to reward feedback, effectively mitigating human subjectivity factors. Compared to DDPG_FP, QMIX_FP demonstrates more pronounced advantages on complex datasets, suggesting the robustness of the multi-agent reinforcement learning algorithm. In summary, the drop in prediction accuracy of QMIX_FP is lower than that of other methods on both the CIFAR-10 and CIFAR-100 datasets, which demonstrates the model's stability.

Ablation study

We conduct ablation studies to analyze different components of the QMIX_FP further. This includes the QMIX algorithm for exploring optimal pruning strategy and the knowledge distillation technology for fine-tuning the pruned networks. Here, we report the results of VGG-16 on CIFAR-10 with prediction accuracy on the test dataset. Similar experimental conclusions can be identified in other deep convolutional neural networks and datasets.

Firstly, the contribution of the network pruning process based on QMIX is shown in Fig. 4. Our proposed pruning method is the “QMIX Prune,” while the “Iterative Prune” is a traditional rule-based pruning method. The horizontal axis in the figure represents the given pruning rate, and the vertical axis represents the test accuracy. The “Base” is the original prediction accuracy of the pre-trained network. Analyzing the model performance of the pruned network, we find that regardless of the pruning rate or the feature importance evaluate function, the test accuracy based on the QMIX prune method is higher than that of the traditional iterative prune method. Notably, the ordinary iterative method prunes a fixed number of filters, but the QMIX-based method tends to

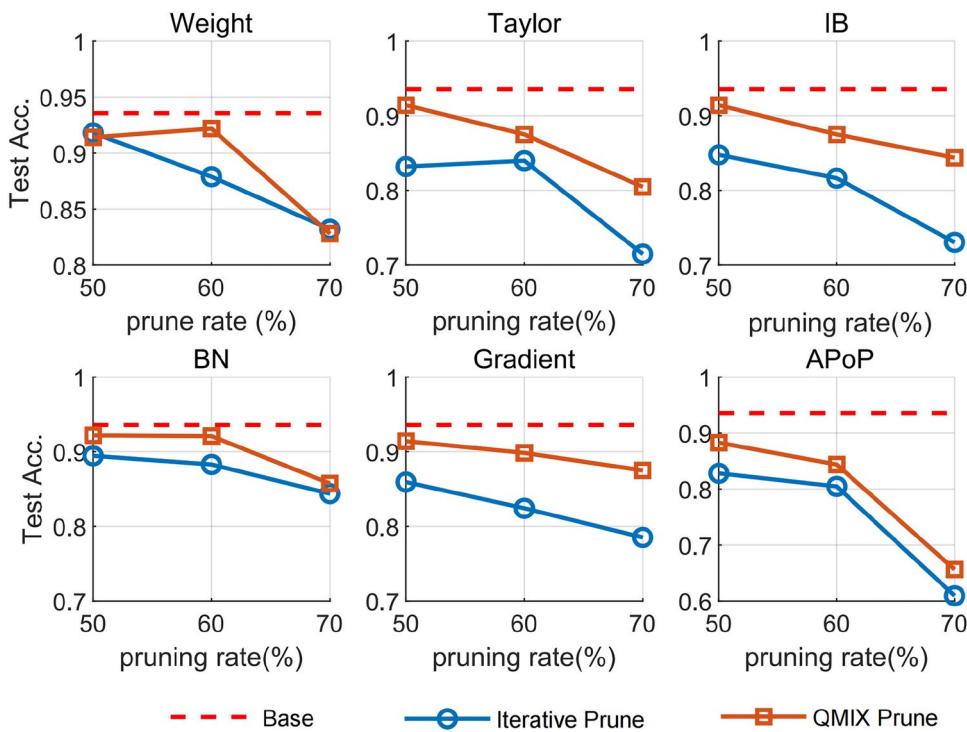


Fig. 4. The ablation study results on the QMIX algorithm.

prune more filters than it. Here, the expected pruning rate is the criterion for determining whether an episode is terminated, whereas the real pruning rate is calculated based on a probability (action). It illustrates that the pruning strategy explored based on the QMIX algorithm is more effective as it prunes more filters while incurring smaller losses in test accuracy.

Secondly, the contribution of the fine-tuning process based on knowledge distillation is shown in Fig. 5. When the temperature parameter T is frozen to 0 in Sect. “Fine-tuning,” it means that the S-Net cannot learn the generalization performance of the T-Net and instead improves model performance only through training itself. To verify the effectiveness of knowledge distillation, we compare our experimental results with the case of $T = 0$. The “Only S_Net” indicates fine-tuning the pruned network by retraining, while “S_Net with KD” is ours in the figure. The horizontal axis represents the training epoch, and the vertical axis represents the test accuracy. It shows that the fine-tuning method based on knowledge distillation converges faster and achieves a test accuracy approximately 1% higher at convergence compared to the retraining-based method.

Conclusions

We propose an automatic filter pruning approach grounded in reinforcement learning, distinct from conventional rule-based or single-agent reinforcement learning-based approaches. Our method, termed QMIX_FPN, leverages the multi-agent reinforcement learning algorithm QMIX to discover the optimal pruning strategy autonomously. This innovation not only streamlines the pruning process but also enhances the efficiency and effectiveness of DCNNs across various applications.

The efficacy of the proposed method was validated by applying it to VGG-16 and AlexNet on the CIFAR-10 and CIFAR-100 datasets, achieving superior performance across multiple benchmarks. QMIX_FPN stands out as a versatile framework for network pruning, adaptable to a wide range of DCNN architectures and datasets. Its flexibility lies in the ease of adjusting the number of agents and state transition functions to align with the specific structural characteristics of the network in question.

The societal impact of QMIX_FPN is significant, particularly in resource-constrained environments such as mobile devices and edge computing platforms. By enabling more efficient and compact neural network models, our method contributes to the advancement of technologies that require real-time processing and minimal power consumption. This is particularly relevant for applications in healthcare, autonomous driving, and environmental monitoring, where the deployment of sophisticated AI models on portable or embedded systems can lead to advances.

Furthermore, our work builds on and complements existing research, such as MR-DCAE⁴³, which uses manifold regularization in deep convolutional autoencoders for unauthorized broadcast identification and a real-time constellation image classification method for wireless communication signals based on the lightweight MobileViT network⁴⁴. These studies highlight the growing trend of integrating advanced machine-learning techniques with specific domain knowledge to solve complex real-world problems. In line with this trend, future research directions for QMIX_FPN include exploring the integration of reinforcement learning with other model compression techniques, such as parameter quantization and low-rank decomposition. In particular, we aim to

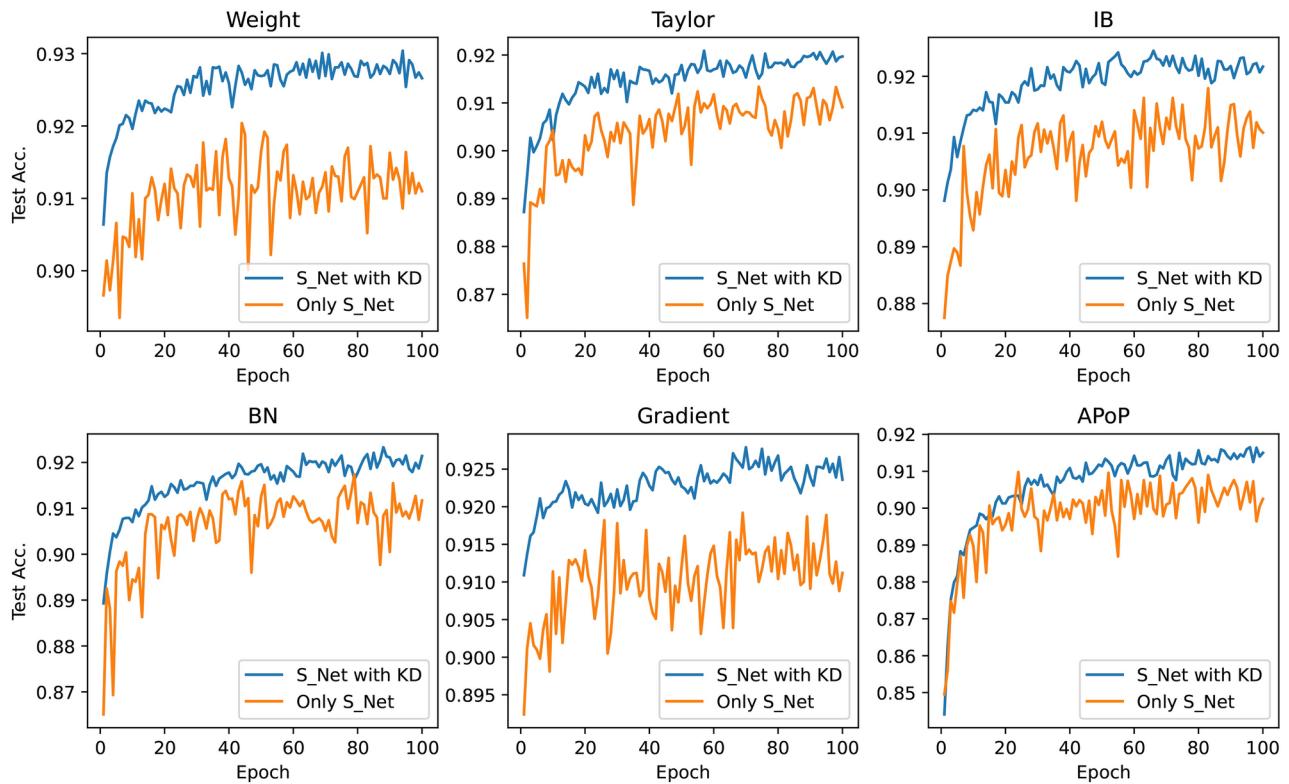


Fig. 5. The ablation study results on knowledge distillation(KD).

investigate the optimal quantization precision for different convolutional features, using the QMIX algorithm to address mixed-precision parameter quantization challenges. Through these efforts, we aim to further improve the practical applicability and efficiency of neural network models in various fields.

Data availability

The datasets used during the current study can be accessed at <https://www.cs.toronto.edu/~kriz/cifar.html> or are available on reasonable request from X. Z. (zuoxiaojinger@163.com).

Received: 5 September 2024; Accepted: 6 December 2024

Published online: 28 December 2024

References

- Krizhevsky, A., Sutskever, I. & Hinton, G. E. Imagenet classification with deep convolutional neural networks. *Commun. ACM* **60**, 84–90 (2017).
- Altaheri, H., Muhammad, G. & Alsulaiman, M. Dynamic convolution with multilevel attention for EGG-based motor imagery decoding. *IEEE Internet Things J.* **10**, 18579–18588 (2023).
- Li, J., Li, Y. & Du, M. Comparative study of EGG motor imagery classification based on dscnn and elm. *Biomed. Signal Process. Control* **84**, 104750 (2023).
- He, K., Zhang, X., Ren, S. & Sun, J. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 770–778 (2016).
- Long, J., Shelhamer, E. & Darrell, T. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 3431–3440 (2015).
- Liu, W. et al. Ssd: Single shot multibox detector. In *Computer Vision-ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11–14, 2016, Proceedings, Part I* 14, 21–37 (2016).
- Sun, J., Li, C., Wang, Z. & Wang, Y. A memristive fully connect neural network and application of medical image encryption based on central diffusion algorithm. *IEEE Trans. Indus. Inform.* (2023).
- Gehring, J., Auli, M., Grangier, D., Yarats, D. & Dauphin, Y. N. Convolutional sequence to sequence learning. In *International Conference on Machine Learning*, 1243–1252 (2017).
- Sun, J., Zhai, Y., Liu, P. & Wang, Y. Memristor-based neural network circuit of associative memory with overshadowing and emotion congruent effect. *IEEE Trans. Neural Netw. Learning Syst.* 1–13 (2024).
- Sun, J., Yue, Y., Wang, Y. & Wang, Y. Memristor-based operant conditioning neural network with blocking and competition effects. *IEEE Trans. Indus. Inform.* **20**, 10209–10218 (2024).
- Abdel-Hamid, O. et al. Convolutional neural networks for speech recognition. *IEEE/ACM Trans. Audio, Speech, Language Process.* **22**, 1533–1545 (2014).
- Dosovitskiy, A. et al. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations* (2020).
- Achiam, J. et al. Gpt-4 Technical Report. *arXiv preprint arXiv:2303.08774* (2023).

14. He, Y. & Xiao, L. Structured pruning for deep convolutional neural networks: A survey. *IEEE Trans. Pattern Anal. Mach. Intell.* **46**, 2900–2919 (2023).
15. Vadera, S. & Ameen, S. Methods for pruning deep neural networks. *IEEE Access* **10**, 63280–63300 (2022).
16. Ding, C. et al. Structured weight matrices-based hardware accelerators in deep neural networks: FPGAs and ASICs. In *Proceedings of the 2018 on Great Lakes Symposium on VLSI*, 353–358 (2018).
17. Kim, N. J. & Kim, H. Fp-agl: Filter pruning with adaptive gradient learning for accelerating deep convolutional neural networks. *IEEE Trans. Multimed.* **25**, 5279–5290 (2023).
18. He, Y. et al. AMC: Automl for model compression and acceleration on mobile devices. In *Proceedings of the European Conference on Computer Vision* **11211**, 815–832 (2018).
19. Feng, Y., Huang, C., Wang, L., Luo, X. & Li, Q. A novel filter-level deep convolutional neural network pruning method based on deep reinforcement learning. *Appl. Sci.* **12**, 11414 (2022).
20. Gupta, M., Aravindan, S., Kalisz, A., Chandrasekhar, V. & Jie, L. Learning to prune deep neural networks via reinforcement learning. *arXiv preprint arXiv:2007.04756* (2020).
21. Rashid, T. et al. Monotonic value function factorisation for deep multi-agent reinforcement learning. In *International Conference on Machine Learning*, 4295–4304 (2018).
22. Hinton, G. et al. Distilling the knowledge in a neural network. *Comput. Sci.* **14**, 38–39 (2015).
23. Bencsik, B. & Szemenyei, M. Efficient neural network pruning using model-based reinforcement learning. In *2022 International Symposium on Measurement and Control in Robotics (ISMCR)*, 1–8 (2022).
24. Kuang, J., Shao, M., Wang, R., Zuo, W. & Ding, W. Network pruning via probing the importance of filters. *Int. J. Mach. Learn. Cybern.* **13**, 2403–2414 (2022).
25. Han, S., Pool, J., Tran, J. & Dally, W. Learning both weights and connections for efficient neural network. *Adv. Neural Inform. Process. Syst.* **28**, 1135–1143 (2015).
26. Han, S., Mao, H. & Dally, W. J. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. *Int. Conf. Learning Represent.* **56**, 3–7 (2016).
27. Yang, T.-J., Chen, Y.-H. & Sze, V. Designing energy-efficient convolutional neural networks using energy-aware pruning. In *IEEE Conference on Computer Vision and Pattern Recognition*, 6071–6079 (2017).
28. Sanh, V., Wolf, T. & Rush, A. Movement pruning: Adaptive sparsity by fine-tuning. *Adv. Neural Inform. Process. Syst.* **33**, 20378–20389 (2020).
29. He, Y., Zhang, X. & Sun, J. Channel pruning for accelerating very deep neural networks. In *IEEE International Conference on Computer Vision*, 1389–1397 (2017).
30. Li, H., Kadav, A., Durdanovic, I., Samet, H. & Graf, H. P. Pruning filters for efficient convnets. In *International Conference on Learning Representations* (2017).
31. Liu, Z. et al. Learning efficient convolutional networks through network slimming. In *IEEE International Conference on Computer Vision*, 2736–2744 (2017).
32. Zhuang, T. et al. Neuron-level structured pruning using polarization regularizer. *Adv. Neural Inform. Process. Syst.* **33**, 9865–9877 (2020).
33. Sarvani, C., Ghorai, M., Dubey, S. R. & Basha, S. S. Hrel: Filter pruning based on high relevance between activation maps and class labels. *Neural Netw.* **147**, 186–197 (2022).
34. Li, G. et al. Optimizing deep neural networks on intelligent edge accelerators via flexible-rate filter pruning. *J. Syst. Architect.* **124**, 102431 (2022).
35. Camci, E., Gupta, M., Wu, M. & Lin, J. Qlp: Deep q-learning for pruning deep neural networks. *IEEE Trans. Circuits Syst. Video Technol.* **32**, 6488–6501 (2022).
36. Molnar, C. *Interpretable Machine Learning* (Lulu. com, 2022), 2 edn.
37. Tishby, N., Pereira, F. C. & Bialek, W. The information bottleneck method. *Clin. Orthop. Related Res. physics/0004057*, 368–377 (2000).
38. Hu, H., Peng, R., Tai, Y.-W. & Tang, C.-K. Network trimming: A data-driven neuron pruning approach towards efficient deep architectures. *arXiv preprint arXiv:1607.03250* (2016).
39. Simonyan, K. & Zisserman, A. Very deep convolutional networks for large-scale image recognition. In *International Conference on Learning Representations* (2014).
40. Krizhevsky, A. et al. Learning multiple layers of features from tiny images. *Handb. Syst. Autoimmune Dis.* **1**, 32–35 (2009).
41. Lin, M. et al. Hrank: Filter pruning using high-rank feature map. In *IEEE Conference on Computer Vision and Pattern Recognition*, 1529–1538 (2020).
42. Basha, S. S., Farazuddin, M., Pulabaigari, V., Dubey, S. R. & Mukherjee, S. Deep model compression based on the training history. *Neurocomputing* **573**, 127257 (2024).
43. Zheng, Q., Zhao, P., Zhang, D. & Wang, H. Mr-dcae: Manifold regularization-based deep convolutional autoencoder for unauthorized broadcasting identification. *Int. J. Intell. Syst.* **36**, 7204–7238 (2021).
44. Zheng, Q. et al. A real-time constellation image classification method of wireless communication signals based on the lightweight network Mobilevit. *Cogn. Neurodyn.* **18**, 659–671 (2024).

Acknowledgements

This work was supported by the National Natural Science Foundation of China [61773020]. The authors would like to express their sincere gratitude to the referees for their careful reading and insightful suggestions.

Author contributions

Z. L. and X. Z. conceived and conducted the experiments. Y. S., D. L., and Z. X. supervised the experimental progress and provided project support. All authors reviewed the manuscript.

Declaration

Competing interests

The authors declare no competing interests.

Additional information

Correspondence and requests for materials should be addressed to Z.X.

Reprints and permissions information is available at www.nature.com/reprints.

Publisher's note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Open Access This article is licensed under a Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 International License, which permits any non-commercial use, sharing, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if you modified the licensed material. You do not have permission under this licence to share adapted material derived from this article or parts of it. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit <http://creativecommons.org/licenses/by-nc-nd/4.0/>.

© The Author(s) 2024, corrected publication 2025