Neural Networks 152 (2022) 407–418

Contents lists available at ScienceDirect

Neural Networks

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

LAP: Latency-aware automated pruning with dynamic-based filter selection

Zailong Chen[a,*], Chubo Liu[a], Wangdong Yang[a], Kenli Li[a], Keqin Li[b]

[a] College of Information Science and Engineering, Hunan University, Hunan 410082, China
[b] Department of Computer Science, State University of New York, New Paltz, NY 12561, USA

ARTICLE INFO

Article history:
Received 20 October 2021
Received in revised form 20 February 2022
Accepted 2 May 2022
Available online 10 May 2022

Keywords:
AutoML
Channel pruning
Model compression and acceleration
Reinforcement learning

ABSTRACT

Model pruning is widely used to compress and accelerate convolutional neural networks (CNNs). Conventional pruning techniques only focus on how to remove more parameters while ensuring model accuracy. This work not only covers the optimization of model accuracy, but also optimizes the model latency during pruning. When there are multiple optimization objectives, the difficulty of algorithm design increases exponentially. So latency sensitivity is proposed to effectively guide the determination of layer sparsity in this paper. We present the latency-aware automated pruning (LAP) framework which leverages the reinforcement learning to automatically determine the layer sparsity. Latency sensitivity is used as a prior knowledge and involved into the exploration loop. Rather than relying on a single reward signal such as validation accuracy or floating-point operations (FLOPs), our agent receives the feedback on the accuracy error and latency sensitivity. We also provide a novel filter selection algorithm to accurately distinguish important filters within a layer based on their dynamic changes. Compared to the state-of-the-art compression policies, our framework demonstrated superior performances for VGGNet, ResNet, and MobileNet on CIFAR-10, ImageNet, and Food-101. Our LAP allowed the inference latency of MobileNet-V1 to achieve approximately 1.64 times speedup on the Titan RTX GPU, with no loss of ImageNet Top-1 accuracy. It significantly improved the pareto optimal curve on the accuracy and latency trade-off.

1. Introduction

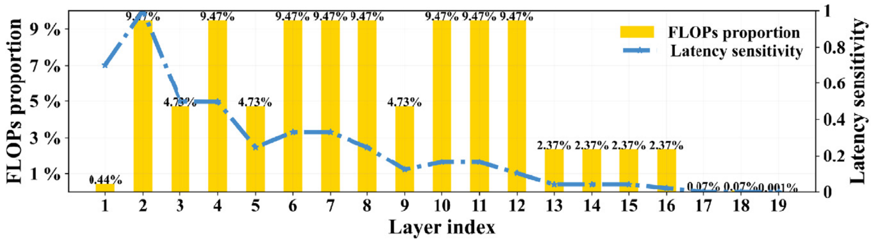
Convolutional neural networks (CNNs) have demonstrated capabilities comparable to or even surpassing humans in many computer vision tasks, such as classification (He, Zhang, Ren, & Sun, 2016; Szegedy et al., 2015), detection (Girshick, Donahue, Darrell, & Malik, 2014; Ren, He, Girshick, & Sun, 2015), and segmentation (Chen, Papandreou, Kokkinos, Murphy, & Yuille, 2014; Long, Shelhamer, & Darrell, 2015). However, CNN is notorious for its huge compute and storage requirements, making it difficult to integrate it into some mobile applications, e.g., smart phones or wearable devices. LeCun, Denker, and Solla (1990) proposed that neural networks have high redundancy, and the unimportant structures can be safely removed without damaging the model performance. This makes it possible to deploy CNN on resource-constrained devices. Hereafter, many effective techniques emerged in the field of model compression and acceleration, which are mainly divided into the following four categories: *parameter pruning and quantization* (Han, Mao, & Dally, 2015; Wu, Leng, Wang, Hu, & Cheng, 2016), *low-rank factorization* (Denton, Zaremba, Bruna, LeCun, & Fergus, 2014; Jaderberg, Vedaldi, & Zisserman, 2014), *transferred/compact convolutional filters* (Shang, Sohn, Almeida, & Lee, 2016; Szegedy, Ioffe, Vanhoucke, & Alemi, 2016), and *knowledge distillation* (Hinton, Vinyals, & Dean, 2015; Romero et al., 2014).

In particular, among model compression technologies, filter pruning aims to abandon the least important filters in CNNs to trade accuracy for latency improvements. It has been proven to be an effective technique and can be divided into two main categories: (1) designing hand-crafted heuristics to determine the pruning policy; (2) using AutoML to search for the optimal substructure. Different layers of CNN have different redundancy, and the filters in a layer also have unequal importance. These make filter pruning face the following two core challenges: (1) how to determine the compression ratio for each layer? (2) how to choose the filters that should be pruned in a layer?

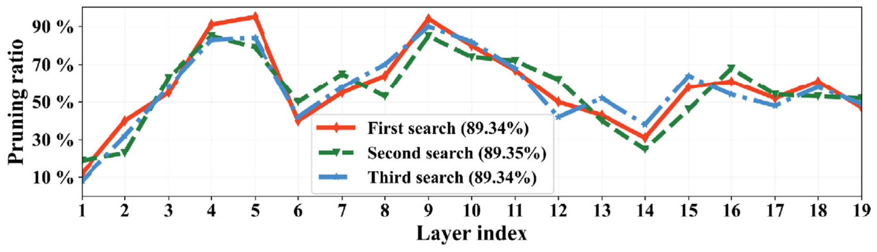
Previously a few researchers have developed alternative approaches to meet the first challenge, including handcraft-based policies (Azarian, Bhalgat, Lee, & Blankevoort, 2020; Lee, Park, Mo, Ahn, & Shin, 2020) and automation-based strategies (He, Lin, et al., 2018; Zhong, Ding, Guo, Han, & Wang, 2018). However,

* Correspondence to: Hunan University, Changsha, Hunan 410082, China.
E-mail addresses: chenzl@hnu.edu.cn (Z. Chen), liuchubo@hnu.edu.cn (C. Liu), yangwd@hnu.edu.cn (W. Yang), lkl@hnu.edu.cn (K. Li), lik@newpaltz.edu (K. Li).

https://doi.org/10.1016/j.neunet.2022.05.002
0893-6080/© 2022 Elsevier Ltd. All rights reserved.



(a) FLOPs proportions and latency sensitivity



(b) Multiple search results

Fig. 1. (a) Different layers have different proportions (bar) of the overall FLOPs and different latency sensitivity (line) in VGG-19 (Simonyan & Zisserman, 2014). (latency sensitivity is normalized into [0, 1]. Pruning a filter in the 2nd convolutional layer yields about 47.5 times FLOPs reduction than that of the 16th convolutional layer.) (b) Multiple search results for VGG-19 on CIFAR-10 (Krizhevsky, Hinton, et al., 2009) under 50% overall sparsity. Searching through reinforcement learning and obtaining different sub-networks with similar accuracy but different FLOP counts.

such methods are all accuracy-oriented. They only consider the different accuracy sensitivity of different layers but ignore their unequal sensitivity to the latency during determining the layer sparsity. Both the accuracy and latency of CNNs can affect the application performance, e.g., mobile devices require high model accuracy and put quite emphasis on processing speed and power consumption. Lower latency means faster inference speed and lower energy consumption, which affect user experience and battery life, respectively.

Input size, kernel size, number of input channels, and number of output channels all affect the count of floating-point operations (FLOPs) of a layer. This results in that different layers tend to have different latency contributions to the model. As shown in Fig. 1(a), shallow layers account for a larger proportion of the overall FLOPs, thus imposing greater sparsity on them can reduce more FLOPs. However, according to the empirical policies (Li, Qian, Jiang, Lu, & Tang, 2018; Morcos, Yu, Paganini, & Tian, 2019), pruning the filters in the shallow layer can cause more model accuracy loss than that of the deep layer. Therefore, the accuracy-oriented pruning approaches may output suboptimal results for model latency. As demonstrated in Fig. 1(b), multiple searches under the same model size constraint yield several sub-networks with similar accuracy but different structures. Because of different latency sensitivity between layers, different structures mean unequal latency despite their parameter number being the same. Fig. 2 (Left) demonstrates the general production process of using prior pruning approaches for finding the high-accuracy and low-latency sub-network of CNN. Step one, using a pruning method to produce a large number of sub-structures with highest accuracy. Step two, comparing these results based on the latency indicator (i.e., FLOPs or inference time) to pick out the fastest one. This pruning-selecting process is quite inefficient and expensive. How to simultaneously optimize accuracy and delay when designing compact models becomes an urgent multi-objective trade-off problem, and it has not yet been explored. This multi-objective optimization is a bigger challenge and makes algorithm design exponentially more difficult. “In order to design a robust solution, it is also necessary to consider factors such as disturbance influence, modeling errors, various uncertainties in real systems, and robustness and filtering techniques mentioned in existing work (Tao, Li, Chen, Stojanovic, & Yang, 2020; Tao, Li, Paszke, Stojanovic, & Yang, 2021; Xin et al., 2022; Xu, Li, & Stojanovic, 2021) should also be considered. Human heuristics or rule-based polices tend to require the domain expertise and have a vast design space. Using such methods to solve this problem is usually high cost, time consuming, labor exhaustive, and may contribute to the suboptimal results. In this work, we would like to automate this exploration process in an end-to-end manner by a learning-based framework.

To this end, we present a latency-aware automated pruning (LAP) framework, which leverages reinforcement learning to automatically predict layer sparsity under arbitrary model size constraint. Latency sensitivity is proposed in this work, which represents the contribution of each layer to the latency. It only incurs a small computational overhead and only needs to be calculated once at the beginning of exploration. It acts as a prior knowledge and is added to the feedback of the exploration process along with the accuracy error. This can avoid balancing the weights between targets when simultaneously optimizing accuracy and latency. More importantly, it effectively prevents multiple iterations of trail-and-error of using accuracy-oriented methods to find the high-accuracy and low-latency model. Specifically, for each layer, the RL agent receives the layer state, and it then outputs the pruning ratio as the action. After all layers are pruned, the network is finetuned for one more epoch. Then, both the accuracy feedback and latency contribution are fed as the reward to our RL agent. For the compression policy within each layer, a novel filter selection algorithm is proposed. The dynamic changes of filters are measured as their importance scores, and those with least changes are pruned first. The principle behind this algorithm is that the more active filter has more adaptability and can compensate for the representation capability of pruned filters. This filter selection method can effectively avoid the exhaustive search by AutoML-based selection approaches (Liu et al., 2019; Yu & Huang, 2019) and the suboptimal results by static-based selection manners (Li, Kadav, Durdanovic, Samet, & Graf, 2016; Lin et al., 2020). The effectiveness of our proposed framework is demonstrated with extensive experiments using VGGNet, ResNet, and MobileNet on CIFAR-10, ImageNet, and Food-101 datasets.

Table 1
Comparison of the pros and cons of various parameter pruning technologies.

Characteristics of parameter pruning technologies		
Inter-layer	Heuristics	Learnable threshold (Azarian et al., 2020) $\cancel{①}$ $\cancel{②}$ $\cancel{③}$, Momentum (Dettmers & Zettlemoyer, 2019) $\cancel{①}$ $\cancel{②}$ $\cancel{③}$, Structure (Evci, Gale, Menick, Castro, & Elsen, 2020) $\cancel{①}$ $\cancel{②}$ $\cancel{③}$.
	AutoML	LSTM (Zhong et al., 2018) $\cancel{①}$ $\cancel{②}$, Reinforcement learning (He, Lin, et al., 2018) $\cancel{①}$ $\cancel{②}$, Binary search (Ding, Ding, Guo, Han, & Yan, 2019) $\cancel{①}$ $\cancel{②}$.
Intra-layer	Heuristics	Feature maps (Lin et al., 2020; Wang, Zhou, Zhang, Bai, & Zhou, 2018) $\cancel{④}$ $\cancel{⑤}$ $\cancel{⑥}$, ℓ_p-norm (He, Kang, Dong, Fu, & Yang, 2018; Li et al., 2016; Zhuo, Qian, Fu, Yang, & Xue, 2018) $\cancel{④}$ $\cancel{⑤}$, Geometric median (He, Liu, Wang, Hu, & Yang, 2019) $\cancel{④}$ $\cancel{⑤}$.
	AutoML	Greedy slimming (Yu & Huang, 2019) $\cancel{⑤}$ $\cancel{⑥}$, Evolutionary algorithm (Liu et al., 2019) $\cancel{⑤}$ $\cancel{⑥}$.
Main characteristics of this paper		
Inter-layer	① End-to-end production process	✓
	② Considering the latency sensitivity of each layer	✓
	③ No need of domain expertise	✓
Intra-layer	④ Accurate filter selection	✓
	⑤ Considering the dynamic changes of filters	✓
	⑥ No time-consuming filter selection process	✓

The main contributions and differences of this paper are listed as follows.

- **Latency-Aware**: The different FLOPs proportions between layers in CNN are shown and analyzed. In addition, latency sensitivity is first proposed and involved into the design of high-performance compact models.
- **Automation**: We present an automated framework for model compression, which searches the sub-network with better accuracy and latency in an end-to-end manner. It frees the human labor from exploring the huge design space and expensive production process.
- **Filter Selection**: A simple yet valid algorithm is offered, which takes the dynamic changes of filters into account during determining the important filters. An effective filter regrowth strategy is also provided to achieve more accurate filter selection.

This paper is organized as follows. Section 2 introduces related work. Section 3 introduces the methodology. Section 4 evaluates the effectiveness of our framework. Section 5 concludes the paper.

2. Related work

Filter pruning is one of the model compression technologies, which aims to compress and accelerate CNNs at the cost of negligible accuracy loss (summarized in Table 1). It can be divided into two categories: *inter-layer-based* policy and *intra-layer-based* strategy.

2.1. Inter-layer-based policy

This policy aims to determine the sparsity ratio of each layer in CNN and can be divided into two types of research methods. *Heuristic-based* approach is one of them, which needs human labor to design hand-crafted rules. For example, (1) Azarian et al. (2020) proposed an algorithm, which learns the pruning threshold of each layer via gradient descent. (2) Dettmers and Zettlemoyer (2019) calculated the normalized momentum of every layer as their importance scores. (3) Evci et al. (2020) computed the pruning ratio of each layer based on their structure parameter. Because this type of techniques tends to suffer from a huge design space and require domain expertise, leading to suboptimal results, many researchers have turned their eyes to the *AutoML-based* methods, which can automatically find the compression policy for each layer. For instance, (1) Zhong et al. (2018) applied long short-term memory (LSTM) to predict which layers can be pruned. (2) He, Lin, et al. (2018) automated the exploration process of layer sparsity via reinforcement learning. (3) Ding et al. (2019) presented an approach to simultaneously search the filter numbers that need to be pruned on multiple layers in a binary search manner. However, aforementioned methods only focus on the accuracy sensitivity of layers, leading to an expensive pruning-selecting production process and suboptimal results when faced multiple design objectives.

2.2. Intra-layer-based strategy

The technique in this category is to select unimportant filters within a layer to prune, which also has two research directions. *Heuristic-based*: (1) Lin et al. (2020) applied the rank of feature maps to guide the filter selection. (2) Wang et al. (2018) searched the key filters by employing subspace clustering on feature maps. These methods are relatively compute-consuming as they select the filters based on feature maps which tend to have a large size. Therefore, many researchers set out to design more efficient filter pruning methods. For instance, (3) Li et al. (2016) calculated the ℓ_1-norm of filters as their importance scores and pruned the least important ones. (4) The ℓ_2-norm of filters are represented as their importance in He, Kang, et al. (2018) and Zhuo et al. (2018). (5) He et al. (2019) computed the geometric median in a layer and pruned the filters closest to this. Nevertheless, the methods mentioned above are all based on the static characteristics of the network and ignore the importance of the filter's dynamic changes, which may result in the loss of key information. For example, two filters with the same ℓ_p-norm may have opposite directions. Static-based methods may not be able to identify the adaptability of the filter when the network is pruned and

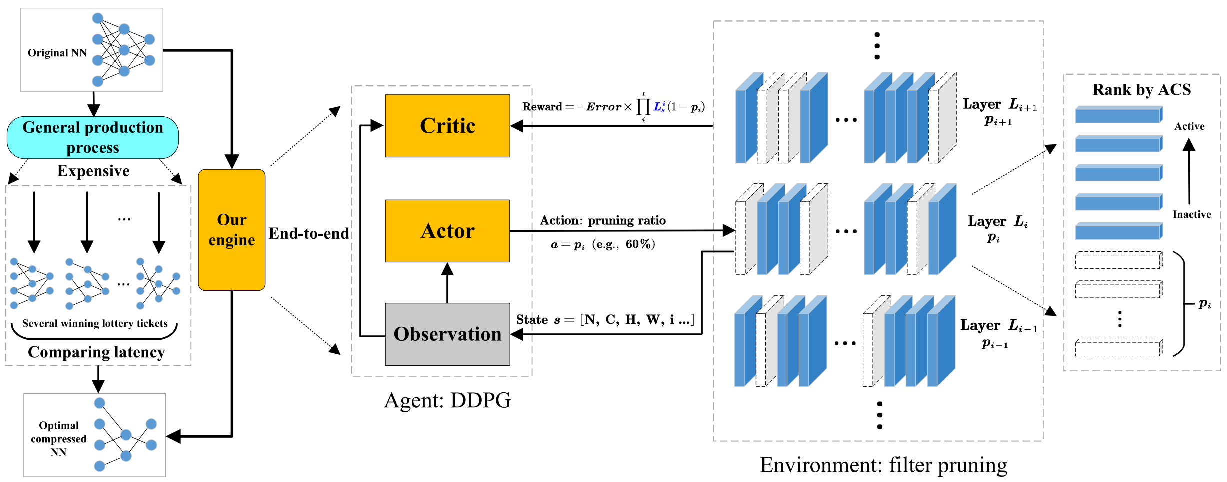

Original NN
General production process
Expensive
Several winning lottery tickets
Comparing latency
Optimal compressed NN
Our engine
End-to-end
Critic
Actor
Observation
Agent: DDPG
Reward = − Error × $\prod_{i}^{l} L_s^i (1-p_i)$
Action: pruning ratio
$a = p_i$ (e.g., 60%)
State s = [N, C, H, W, i ...]
Layer L_{i+1} p_{i+1}
Layer L_i p_i
Layer L_{i-1} p_{i-1}
Environment: filter pruning
Rank by ACS
Active
Inactive
p_i

Fig. 2. Overview of our method. Left: Our method replaces the expensive pruning-selecting process of the general production of compact models. Middle: Form layer-adaptive sparsity as a reinforcement learning problem. Our agent process a pre-trained CNN in a layer-by-layer manner. it receives the state s_i from layer L_i and then outputs a pruning ratio p_i as the action a. After the pruned network is finetuned, the combination of accuracy error and latency sensitivity is fed to our RL agent as the reward. Right: When the RL agent outputs the pruning ratio p_i of layer L_i, the adjusted cosine similarity (ACS) of the filters in this layer are calculated as their importance. The least important filters are pruned first.

becomes incomplete. Moreover, a few *AutoML-based* pruning approaches have been proposed. For example, (1) Yu and Huang (2019) proposed to train a slimmable network and greedily slim the layer with minimal accuracy drop. (2) Liu et al. (2019) trained a meta network and used an evolutionary procedure to search the least important channels. However, exhaustively searching the least important filters in a network by AutoML-based methods is computationally prohibitive. Considering a network with ten layers and each layer contains 64 channels. Under a uniform 50% pruning ratio, the possible combination of pruning could be $(C_{64}^{32})^{10}$, which is an intolerable search space.

2.3. Discussion

To this end, for the layer-adaptive sparsity, our framework automates this exploration process and takes the latency sensitivity ignored by the previous works into account. It can search the compression ratio of each layer in an end-to-end manner and better trade off the accuracy and latency. In terms of the compression policy within a layer, this work focuses on the dynamic-based filter selection. The filter's dynamic changes that overlooked by other heuristics are measured, and the least active ones are pruned according to their importance scores under the pruning ratio given by our RL agent.

3. Methodology

We model the exploration of the layer sparsity as a reinforcement learning problem, and propose a new filter selection method which rank the filters by calculating their adjusted cosine similarity (ACS) during the search process (Fig. 2). In our framework, the actor–critic model with the agent of deep deterministic policy gradient (DDPG) (Lillicrap et al., 2015) is applied to provide the action: pruning ratio of each layer. The combination of accuracy error and latency contribution is used as the reward to search the optimal compression policy. The details of our framework are described below.

3.1. Preliminary

CNN can be represented by the trainable weights $\mathbf{W}=\left\{\mathbf{W}^i \in \mathbb{R}^{O_i \times I_i \times K_i \times K_i},\ 1 \le i \le l\right\}$, where $\mathbf{W}^i$ is the connection weights of the ith weighted layer, which connects the layer L_i and layer L_{i+1}. O_i and I_i are the number of output channels and the number of input channels for layer L_i. K_i represents the kernel size and l is the number of the weighted layers. The shapes of input tensor $\mathbf{U}$ and output tensor $\mathbf{V}$ are $I_i \times H_i \times W_i$ and $O_i \times H_{i+1} \times W_{i+1}$, respectively. The convolutional operation and FLOPs related calculation of the ith layer can be represented as:

$$\mathbf{V}_j^i = \mathcal{F}_j^i \times \mathbf{U} \ \mathit{for}\ 1 \le j \le O_i, \tag{1}$$

$$R_f^i = 2H_iW_i(I_iK_i^2+1) + 2H_{i+1}W_{i+1}K_{i+1}^2O_{i+1} \tag{2}$$

where $\mathcal{F}_j^i \in \mathbb{R}^{I_i \times K_i \times K_i}$ and $\mathbf{V}_j^i$ denote the jth filter and the jth output feature map of the ith layer, respectively. $\mathbf{W}^i$ consists of $\left\{\mathcal{F}_j^i,\ 1 \le j \le O_i\right\}$. R_f^i is the number of FLOPs reduction of pruning a filter in the ith convolutional layer. Of note is that removing output channel in layer L_i will lead to the reduction of input channel in layer L_{i+1}.

3.2. Observation (state space)

Our agent acts on the network layer by layer. We use the same state space for every layer. For each layer L_i, the state s^i is formulated as:

$$(i, N, I, O, H, W, stride, K, L_s^i, N_{params}, a_{i-1}) \tag{3}$$

where i is the layer index, the dimension of the filter is $I \times K \times K$, and the input size is $I \times H \times W$. The number of all weights in layer i are $I \times O \times K \times K$. L_s^i denotes the latency sensitivity of the ith layer, which is normalized into [0, 1] by the R_f^i in Eq. (2). N_{params} is remaining parameter number of layer L_i. a_{i-1} denotes the action from the last time step. The agent can distinguish one convolutional layer from another by these states.

In addition, our agent only needs to know how much accuracy loss and FLOPs reduction the network incurs in each iteration of the optimization process. The accuracy of the network is validated at the end of each epoch and fed back directly to the agent. And the FLOPs reduction of layer L_i can be computed by multiplying its pruning ratio by L_s^i. These essential features consist the state space.

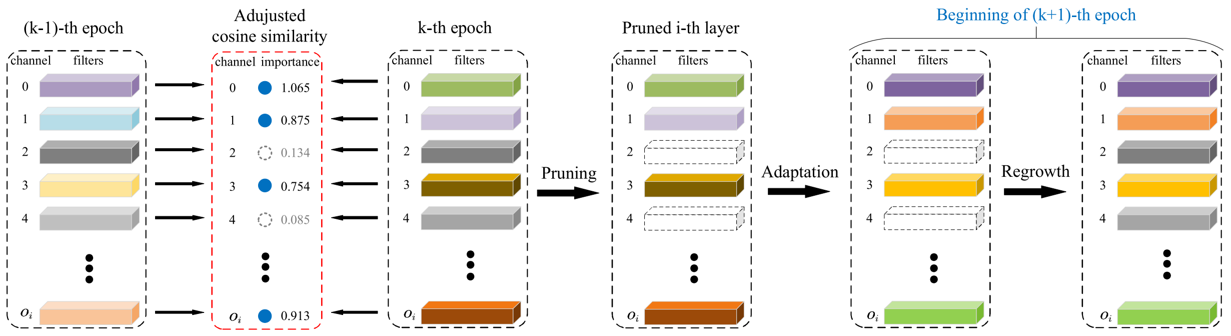


Fig. 3. Overview of the calculation of ACS, pruning, adaptation, and regrowth. At the first of kth epoch, the ACS of the filters in every weighted layer between two epochs are measured as their importance evaluation. Then the filters are ranked by the ACS (red dotted rectangle) and the smaller ones (gray dotted circle) are selected to be zeroed (black dotted cuboid). After pruning, the network is finetuned for one epoch to recover the performance. Then the pruned filters are regrowed. This operation is performed at the beginning of the $(k+1)$-th epoch. It reinitializes the pruned filters to their unpruned values for the next round of importance evaluation of filters (i.e., 2nd and 4th filters for $(k+1)$-th epoch are regrowed from zero to their values for kth epoch).

3.3. Action space

"Model compression is quite sensitive to the sparsity ratio, and discrete and coarse-grained action spaces may result in suboptimal results. If a fine-grained action space is used, the number of discrete actions explodes as the network gets deeper and deeper. It is difficult to explore effectively in such a large action space (Lillicrap et al., 2015). Furthermore, the discretization departs from the relative order, e.g., 10% sparsity is more aggressive than 20% and even more aggressive than 30%. Therefore, we utilize a continuous action space $a \in [0, 1]$ to achieve more fine-grained and accurate compression". By limiting the action space, the preset model size constraint can be accurately achieved (He, Lin, et al., 2018).

3.4. Filter selection

Adaptability of filter. Under the action a (pruning ratio) given by our agent, we measure the dynamic changes of filters in the corresponding layer as their importance scores. Then the least important ones are pruned first (Fig. 3). Because euclidean distance can only measure the distance between data but not the difference in their directions. On the contrary, cosine similarity distinguishes the difference from the direction, and is not sensitive to the magnitude. It is necessary to use a metric that can effectively capture changes in direction and magnitude, and the ACS is an ideal choice. Its computational complexity is also low, thus bringing about negligible time overhead. In order to compute the filters' ACS, $\mathbf{W}^i$ is reshaped into a new weight matrix $\mathcal{M}^i$ with dimension $[O_i, I_i \times K_i \times K_i]$, where O_i is the number of filters, $I_i \times K_i \times K_i$ is the number of weights of a filter. Without confusion, $\mathcal{F}^i$ is still used to denote the filter vector in $\mathcal{M}^i$ weight matrix during calculating the ACS. Specifically,

$$\bar{\mathcal{F}}^i_k = \frac{\sum_{j=1}^{O_i} \mathcal{F}^i_{j,k-1} + \sum_{j=1}^{O_i} \mathcal{F}^i_{j,k}}{2O_i} \tag{4}$$

$$\mathcal{F}^i_j = \mathcal{F}^i_j - \bar{\mathcal{F}}^i, \tag{5}$$

where $\mathcal{F}^i_{j,k-1}$ denotes the jth filter of layer L_i for the $(k-1)$-th epoch. $\bar{\mathcal{F}}^i$ is the mean of the $2O_i$ filters in layer L_i for the $(k-1)$-th and kth epochs. Then, each filter in $\mathcal{M}^i_{k-1}$ and $\mathcal{M}^i_k$ minuses $\bar{\mathcal{F}}^i$ (see Eq. (5)). This is the adjusted step of ACS. Next, the ACS of each filter is calculated, details are shown below:

$$\mathcal{S}^i \in \mathbb{R}^{O_i} = \left\{ \mathcal{C}^i_{j,k} = 1 - \frac{\mathcal{F}^i_{j,k-1}\mathcal{F}^i_{j,k}}{\|\mathcal{F}^i_{j,k-1}\| \, \|\mathcal{F}^i_{j,k}\|}, 1 \le j \le O_i \right\}, \tag{6}$$

where $\mathcal{C}^i_{j,k}$ is the ACS of the jth filter in layer L_i between the $(k-1)$-th epoch and the kth epoch, using one minus the ACS is to fit the nature definitions of similarity, that is, when a filter has no change, it is zero, and a filter with larger changes has a larger value. $\mathcal{S}^i$ consists of the ACS of all filters in layer L_i. The filter with larger ACS is more active and more adaptable to the destruction of the network structure. When the network is pruned, more active filters can quickly make adjustments to compensate for the representation capabilities of the pruned filter.

Pruning. In line with the aforementioned understanding, such filters with smaller ACS should be pruned in preference to those with larger ACS. Specifically, according to the pruning ratio p_i given by our agent, $O_i p_i$ unimportant filters in layer L_i are selected for pruning, i.e., the gray filters in Fig. 3.

Adaptation. After all pruning steps, the network is finetuned for one more epoch to adapt the remaining filters to the pruned network (see the adaptation step of Fig. 3). After this step, the validation accuracy of the pruned network can be obtained to detect whether the selected filters can effectively recover the representation ability of the model.

Regrowth. The reinforcement learning framework needs a lot of search iterations before finding the reasonable layer sparsity. During the search, the importance of filters changes as the pruned network is fine-tuned. Therefore, those pruned filters need to be regrowed to participate in the next round of importance evaluation. As shown in Fig. 3, we regrow the pruned filters to their unpruned state at the beginning of the $(k+1)$-th epoch. Then, the network is retrained for one epoch to adjust all filters into a new state for selection. The effectiveness of this policy is empirically demonstrated in Section 4.3.

3.5. Reward function

Most of the existing methods use a single signal as the reward (e.g., accuracy, FLOPs, or model size), and they optimize a certain objective (e.g., accuracy or FLOPs) under another constraint (e.g., model size). However, this will lead to a local optima for the preset objective, and cannot cover other equally important goals. If the reward is designed to optimize multiple objectives, it will involve balancing the weights of between targets, which is time-consuming and difficult. In addition, it is inevitable to find the appropriate weights again when changing the search model.

As a result, to avoid local optima and weight design, latency sensitivity is introduced as a prior knowledge in this work, and it is involved into the exploration loop to guide the efficient search. After the adaptation step and before the regrowth step of filter selection, the validation accuracy of the pruned network is obtained. As we have already imposed the model size constraint by limiting the action space, the reward is defined to be related to both the accuracy and latency. The agent is encouraged to find the better accuracy and latency by adjusting the sparsity ratio of each layer. Driven by this, the reward function $\mathcal{R}$ is designed as follows:

$$\text{“}\mathcal{R} = -(Acc_{origin} - Acc_{prune}) \times \prod_{i}^{l} L_s^i(1-p_i)\text{”} \tag{7}$$

where Acc_{origin} and Acc_{prune} are the Top-1 accuracy of original model and the pruned model after finetuning, respectively. p_i denotes the pruning ratio of layer L_i. The latency sensitivity L_s of each layer can be obtained by normalizing their FLOPs reduction in Eq. (2) into [0, 1]. The sensitivity equal to zero is modified to 0.001. This reward function is sensitive to the accuracy feedback, it also provides an incentive for FLOPs reduction. RL agent can obtain more positive feedback when it gives high pruning ratios to layers with high latency sensitivity. The latency sensitivity of CNN only needs to be calculated once, thus bringing negligible compute overhead.

3.6. Agent

As demonstrated in Fig. 2, we apply the DDPG, an off-policy actor–critic algorithm for continuous control, in our RL agent to tackle the compression ratio. This agent receives a state s_i of layer L_i and then outputs a pruning ratio as action a_i. Then it moves to the next layer L_{i+1} for processing. After pruning all layers, the reward including both the accuracy error and latency sensitivity is fed to the agent.

During the exploration noise process, the following stochastic process is used and shown below:

$$w'(s_i) \sim \mathcal{N}_{\text{trunc}}\left(w\left(s_i \mid \theta_i^w\right), \sigma^2, 0, 1\right) \tag{8}$$

where $\mathcal{N}_{\text{trunc}}(\mu, \sigma, a, b)$ denotes the truncated normal distribution, and w is model weights. The noise σ is initialized to 0.5 and is decayed exponentially after each episode. The truncated normal distribution is used here to prevent excessive noise and to limit it to the [0, 1] range.

A variant form of Bellman's Equation is employed, where each transition in an episode is $(s_i, a_i, \mathcal{R}, s_{i+1})$. During the update, the Q-function is:

$$\hat{Q}_i = \mathcal{R}_i - \mathcal{B} + \gamma \times Q\left(s_{t+1}, w\left(s_{t+1}\right) \mid \theta^Q\right) \tag{9}$$

and the loss function is presented as:

$$\mathcal{L} = \frac{1}{N}\sum_{t=1}^{N}\left(\hat{Q}_t - Q\left(s_t, a_t \mid \theta^Q\right)\right)^2 \tag{10}$$

where N represents the number of steps in this episode, and the baseline reward $\mathcal{B}$, defined as an exponential moving average of the previous rewards, is to reduce the variance of the gradient estimation. The discount factor γ is set to 1 to prevent excessive emphasis on short-term rewards (Baker, Gupta, Naik, & Raskar, 2016).

4. Experiments

The DDPG (Lillicrap et al., 2015) agent has an actor network and a critic network. They both use the same network architecture: consisting of two hidden layers, each with 300 units. The actor network has an additional sigmoid layer to project the actions into [0,1]. Note that the maximum/minimum sparsity ratios a_{max}/a_{min} can be set as an other value to accelerate the search process, e.g., 0.8 for a_{max} and 0.2 for a_{min}. Extensive experiments are conducted to show the effectiveness and efficiency of our framework.

Datasets and Baselines. Experiments are performed on the datasets of CIFAR-10 (Krizhevsky et al., 2009), Food-101 (Bossard, Guillaumin, & Van Gool, 2014), and ImageNet (Deng et al., 2009). The performance of different algorithms is investigated on widely-used CNN models, including VGGNet (Simonyan & Zisserman, 2014) with plain structure, ResNet (He et al., 2016) with residual blocks, and MobileNet (Howard et al., 2017) with depth-wise and point-wise structures.

Configuration. We employ the deep learning framework Pytorch (Paszke et al., 2017) to implement the proposed LAP framework. All experiments are executed on four NVIDIA TITAN RTX GPUs. According to previous experience (Ding et al., 2019; Luo & Wu, 2020; Luo, Wu, & Lin, 2017) we only prune the internal layers (except for the layers directly added to the feature maps in each block) on ResNet. We keep the Batch Normalization (Ioffe & Szegedy, 2015) layers during pruning instead of merging them into convolutional layers. The maximum sparsity ratios a_{max} for all layers are set to 0.85, and the a_{min} is set to 0.2. Note that modifying the a_{max} and a_{min} is only for faster search, one can simply use a_{max} = 1 which also produces similar results. We use τ = 0.01 for the soft target updates and train the network with 64 as batch size and 2000 as replay buffer size. Our agent first explores 100 episodes with a constant noise σ = 0.5, and then exploits 300 episodes with exponentially decayed noise σ”.

Evaluation Protocols. We employ commonly used protocols, i.e., using model accuracy, required FLOPs count, and inference time to evaluate algorithm effectiveness, computational requirements, and model performance, respectively. In order to evaluate the execution efficiency of our algorithm, the execution time required to optimize each network is presented.

4.1. Results on CIFAR-10

The performance of our framework is evaluated against two inter-layer-based policies (a handcraft-based method RigL Evci et al., 2020 and an automation-based technique AMC He, Lin, et al., 2018) on some popular CNNs, including VGG-16, ResNet-56, and MobileNet-V1. Several experiments with channel pruning are conducted on CIFAR-10. These two techniques are introduced as follows:

- **RigL (handcrafted)** (Evci et al., 2020). This heuristic-based method employs the structure parameter to calculate the layer sparsity, detail is shown below:

$$s^l = 1 - \frac{n^{l-1} + n^l + w^l + h^l}{n^{l-1} \times n^l \times w^l \times h^l} \tag{11}$$

  where s^l and n^l are the pruning ratio and the neuron number of layer L_i, respectively. And w^l and h^l denote the width and height of the convolutional kernel. This method allocates higher sparsity to the layers with more parameters. For pruning policy within a layer, the weight's magnitude is used as importance score. Since this approach aims at weight pruning while our framework focus on filter pruning, we modify the weight selection into the filter selection, that is to rank the filters based on the sum of magnitude of their weights.
- **AMC (automated)** (He, Lin, et al., 2018). This work explores the pruning ratio of each layer via reinforcement learning, and selects the unimportant filters in a layer based on the comparison of their magnitudes.

Latency-Aware Detection. We conduct experiments on VGG-16 and compare our approach with five different compression policies under 50% overall sparsity ratio (see Fig. 4). It is worth mentioning that the gap of accuracy loss caused by different

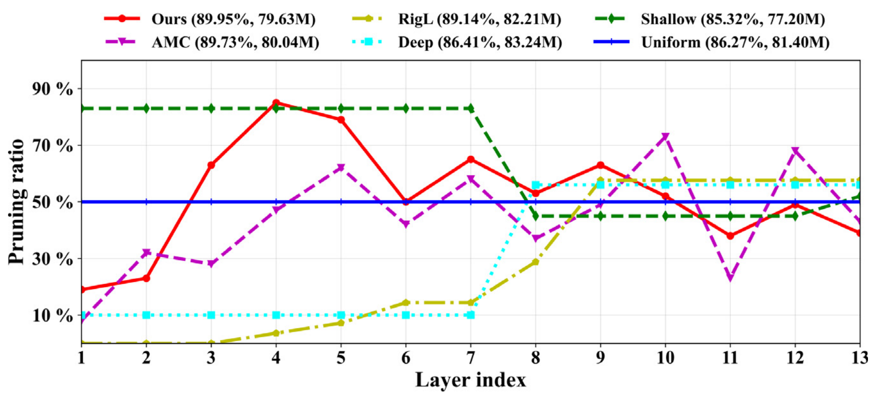


Fig. 4. Comparisons of pruning policies for VGG-16 (Simonyan & Zisserman, 2014) at 50% overall sparsity ratio on CIFAR-10 (Krizhevsky et al., 2009). The compact model produced by our method is superior to other strategies both in accuracy and latency.

Table 2
Pruning method comparison of VGG-16, ResNet-56 (He et al., 2016), MobileNet-V1 (Howard et al., 2017) on CIFAR-10. Infer represents the inference time of pruned models. ExeT is the execution time required to compress the network. M means million (10^6), and GPUh indicates how many hours the optimization process takes when using one GPU.

Network	Method	Top1-Acc	FLOPs	Infer (ms)	ExeT (GPUh)
VGG-16	0.5 × base	86.27%	81.40M (26.00%)	0.73	–
(91.26%,	RigL (Evci et al., 2020)	89.14%	84.21M (26.90%)	0.76	–
313M,	AMC (He, Lin, et al., 2018)	89.73%	79.04M (25.25%)	0.71	–
2.82 ms)	**Ours**	**89.95%**	**75.01M (23.96%)**	**0.68**	**4.43**
ResNet-56	0.5 × base	90.47%	31.40M (25.14%)	0.40	–
(93.26%,	RigL (Evci et al., 2020)	91.05%	32.62M (26.11%)	0.42	–
124.9M,	AMC (He, Lin, et al., 2018)	91.17%	30.34M (24.29%)	0.39	–
1.60 ms)	**Ours**	**91.72%**	**29.01M (23.22%)**	**0.37**	**10.59**
MobileNet-V1 (91.58%, 46M, 5.69 ms)	0.75 × base	88.07%	26.50M (57.60%)	3.28	–
	RigL (Evci et al., 2020)	90.03%	27.53M (59.83%)	3.40	–
	AMC (He, Lin, et al., 2018)	91.17%	25.83M (56.14%)	3.19	–
	Ours	**91.31%**	**24.39M (53.01%)**	**3.02**	**6.84**
	0.5 × base	87.51%	12.10M (26.30%)	1.50	–
	RigL (Evci et al., 2020)	89.42%	12.57M (27.32%)	1.55	–
	AMC (He, Lin, et al., 2018)	89.82%	11.73M (25.49%)	1.45	–
	Ours	**90.09%**	**11.14M (24.21%)**	**1.38**	**7.15**

methods cannot be clearly captured by using small pruning ratios. In addition, some methods can severely damage the network and cannot recover model performance under large pruning ratios, such as deep and shallow policies, making it difficult for evaluating the effectiveness and differences of several methods. So a pruning ratio of 50% is used here. Note that *uniform* sets pruning rate for each layer uniformly, *shallow* (Li et al., 2016) and *deep* (He, Zhang, & Sun, 2017) prune shallow and deep layers respectively. Considering that fully-connected (FC) layers occupy about 89% of the model parameters but less than 1% of the model FLOPs for VGG-16. Therefore, under a preset model size constraint, pruning too much or too little to FC layers will cause huge fluctuations in the pruning ratio of convolutional layers. This makes it difficult to investigate the impact of different pruning methods on the model latency. For example, when the pruning ratio of FC layers is lower than 44%, even if all the convolutional layer parameters are removed, the overall sparsity 50% cannot be achieved. Therefore, here the pruning rate of FC layers is uniformly set to 50%. As illustrated in Fig. 4, our Latency-Aware framework can find the structure with better accuracy and latency compared to other five policies. The model learned by AMC (He, Lin, et al., 2018) has high accuracy but is less latency-friendly. RigL (Evci et al., 2020) tends to prune the layers with more parameters. *Deep* and *shallow* policies result in a relatively serious accuracy loss.

Quick Sanity Check. Here we use VGG-16, ResNet-56, and MobileNet-V1 under different overall sparsity for a quick sanity check. We compare our framework with two inter-layer-based policies: AMC (He, Lin, et al., 2018) and RigL (Evci et al., 2020). Table 2 shows the effectiveness of different approaches. Our framework LAP provides significantly better accuracy and latency. Specifically, for VGG-16, under the overall sparsity 50%, the pruned model obtained by LAP achieves a higher accuracy (89.95% vs. 89.73% by AMC (He, Lin, et al., 2018), and 89.14% by RigL Evci et al., 2020) and less FLOPs (75.01M vs. 79.04M by AMC He, Lin, et al., 2018, and 84.21M by RigL Evci et al., 2020). For ResNet-56 and MobileNet-V1, our framework also outputs more accurate and fast compact models under different model size constraints. Moreover, the inference time of each network compressed by our method is significantly less than other approaches under different sparsities. And the time consumption of our algorithm for the optimization process of each network is also offered in Table 2. In addition, the experimental results show that the automation-based method can find compact models with higher performance compared to the heuristic-based approach.

4.2. Results on ImageNet

We also explore the performance of our framework for VGG-16, ResNet-50, and MobileNet-V1 on ImageNet, and compare it with AMC (He, Lin, et al., 2018), RigL (Evci et al., 2020) and two global channel pruning techniques (Chin et al., 2020; Lee et al., 2020) under different compression ratios. The pruning ratios of 25% and 50% are utilized to check the sensitivity of algorithm outcomes to different sparsity. Here these two global pruning techniques are briefly introduced:

Table 3
For comparison experiments on ImageNet, our method consistently outperforms other pruning techniques both in accuracy and latency. M/B means million/billion ($10^6/10^9$), respectively.

Network	Method	Top1-Acc	FLOPs	Infer (ms)	ExeT (GPUh)
VGG-16 (68.34%, 15.3B, 138 ms)	0.5 × base	61.25%	3.98B (26.01%)	35.90	–
	RigL (Evci et al., 2020)	65.93%	4.14B (27.06%)	37.34	–
	AMC (He, Lin, et al., 2018)	66.52%	3.86B (25.22%)	34.82	–
	LAMP (Lee et al., 2020)	66.40%	3.89B (25.42%)	35.09	–
	LeGR (Chin, Ding, Zhang, & Marculescu, 2020)	66.71%	3.87B (25.29%)	34.91	–
	Ours	**66.93%**	**3.66B (23.92%)**	**33.01**	**18.37**
ResNet-50 (76.60%, 4.1B, 53 ms)	0.75 × base	74.80%	2.30B (56.10%)	29.73	–
	RigL (Evci et al., 2020)	75.86%	2.39B (58.29%)	30.90	–
	AMC (He, Lin, et al., 2018)	76.15%	2.23B (54.39%)	28.83	–
	LAMP (Lee et al., 2020)	75.93%	2.26B (55.12%)	29.21	–
	LeGR (Chin et al., 2020)	76.25%	2.28B (55.61%)	29.47	–
	Ours	**76.36%**	**2.12B (51.70%)**	27.40	36.41
	0.5 × base	72.00%	1.10B (26.82%)	14.22	–
	RigL (Evci et al., 2020)	74.41%	1.14B (27.80%)	14.74	–
	AMC (He, Lin, et al., 2018)	74.88%	1.07B (26.09%)	13.83	–
	LAMP (Lee et al., 2020)	74.65%	1.08B (26.34%)	13.96	–
	LeGR (Chin et al., 2020)	74.99%	1.06B (25.85%)	13.70	–
	Ours	**75.08%**	**1.01B (24.63%)**	**13.06**	**43.74**
MobileNet-V1 (70.60%, 569M, 124 ms)	0.75 × base	68.40%	325M (57.11%)	70.83	
	RigL (Evci et al., 2020)	69.47%	338M (59.40%)	73.66	–
	AMC (He, Lin, et al., 2018)	70.13%	315M (55.36%)	68.65	–
	LAMP (Lee et al., 2020)	69.89%	318M (55.89%)	69.30	–
	LeGR (Chin et al., 2020)	70.18%	317M (55.71%)	69.08	–
	Ours	**70.34%**	**301M (52.90%)**	**65.60**	**26.56**
	0.5 × base	63.70%	149M (26.18%)	32.47	
	RigL (Evci et al., 2020)	68.35%	155M (27.24%)	33.78	–
	AMC (He, Lin, et al., 2018)	68.67%	144M (25.31%)	31.38	–
	LAMP (Lee et al., 2020)	68.53%	145M (25.48%)	31.60	–
	LeGR (Chin et al., 2020)	68.78%	144M (25.31%)	31.36	–
	Ours	**69.02%**	**138M (24.25%)**	**30.07**	**29.72**

- **LAMP (handcrafted)** (Lee et al., 2020) is a magnitude-based pruning algorithm that can learn a global rank of all weights in a network. When obtaining the global rank, the weights/channels with smallest scores are pruned until the required global sparsity limit is met. This procedure is equivalent to the methods of automatically selecting layer sparsity. For a fair comparison, we modify the LAMP (Lee et al., 2020) into a channel pruning, the sum of the weight scores in the filter is calculated as its importance.
- **LeGR (automated)** (Chin et al., 2020). This approach sets a weight pair ($\alpha - \kappa$) at each layer to determine their sparsity, and uses evolutionary algorithms to learn the optimal $\alpha - \kappa$ pairs. The rank of the filters within layers is based on their ℓ_2-norm.

Different Compression Policies. We evaluate the effectiveness of our LAP by comparing it with four prior art on ImageNet. As shown in Table 3, LAP exceeds its counterparts in all aspects, including Top-1 accuracy and FLOPs reduction. More specifically, for ResNet-50 and 75% overall sparsity, our framework outperforms other four methods both on accuracy (76.36% vs. 75.86% by RigL Evci et al., 2020, 76.15% by AMC He, Lin, et al., 2018, 75.93% by LAMP Lee et al., 2020, and 76.25% by LeGR Chin et al., 2020) and FLOP count (2.12B vs. 2.39B by RigL Evci et al., 2020, 2.23B by AMC He, Lin, et al., 2018, 2.26B by LAMP Lee et al., 2020, and 2.28B by LeGR Chin et al., 2020). In addition, for MobileNet-V1, our LAP allows the pruned model to achieve 70.34% Top-1 accuracy and 52.90% FLOP counts of the original network, which significantly surpass the results produced by other four methods by a large margin. Under a more aggressive overall sparsity — 50%, LAP also provides more accurate and faster models for VGG-16, ResNet-50, and MobileNet-V1 against prior art. These results demonstrate the superiority of our framework, which can learn the better pruning ratio for each layer to achieve the faster network with less accuracy loss under arbitrary model size constraint.

Inference Speedup. Recently, inference acceleration has drawn a lot of attention both in industry and academia. By improving the inference latency, not only the processing speed can be increased, but also the energy consumption can be optimized. Therefore, here we evaluate how much the inference speed of MobileNet-V1 can be improved by our framework on ImageNet compared to LAMP (Lee et al., 2020) and LeGR (Chin et al., 2020). As illustrated in Fig. 5(a), our framework significantly improves the pareto optimal curve of MobileNet on the accuracy and MAC trade-off, and surpasses LAMP (Lee et al., 2020) and LeGR (Chin et al., 2020). Specifically, LAP can reduce more FLOPs under similar Top-1 accuracy than other two methods, e.g., 47.10% FLOPs reduction (1-52.90%) vs. 44.29% by LeGR (Chin et al., 2020), and 34.45% by LAMP (Lee et al., 2020) under about 70.20% Top-1 accuracy. As shown in Fig. 5(b), the inference speed of MobileNet is greatly improved by our framework. When compared with other two methods, LAP better trades off the accuracy and latency. Under the same Top-1 accuracy — 70.6%, inference time of the model pruned by LAP is 76 ms. It allows unpruned MobileNet (125 ms) to achieve approximately 1.64 times speedup.

4.3. Ablation study

To comprehensively understand our framework, extensive ablation experiments are conducted on VGGNet, ResNet, and MobileNet with the dataset CIFAR-10, ImageNet, and widely used fine-grained dataset Food-101, which contains 101,000 images of 101 different food categories (75,750/25,250 images for training/test, respectively).

Detection of each key component. We further evaluate the effectiveness of each component in our framework by shielding them separately and observe their impact on model performance. Extensive experiments are performed on VGG-16, ResNet-50, and MobileNet-V1. We compare our latency-aware

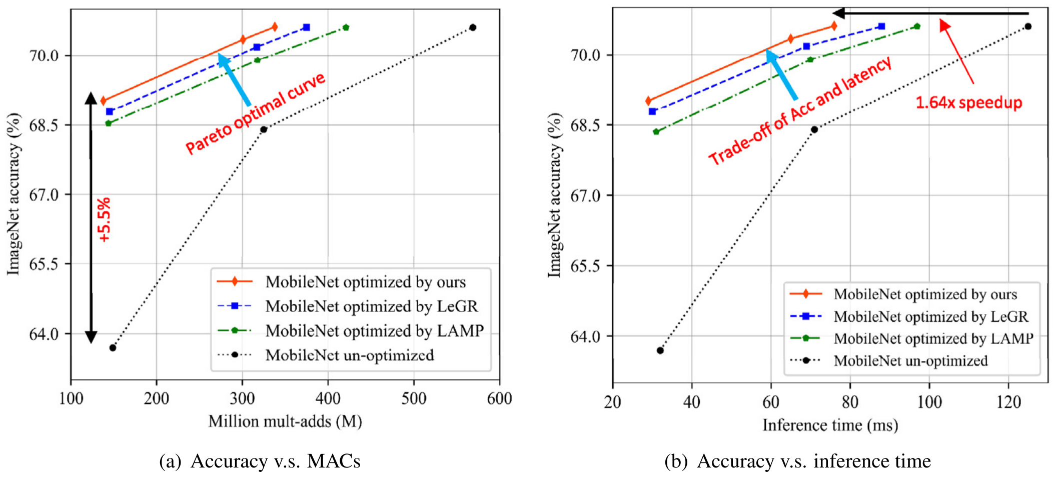


Fig. 5. (a) Comparisons of the accuracy and MAC trade-off among our framework, LeGR (automated) (Chin et al., 2020), LAMP (handcrafted) (Lee et al., 2020), and original MobileNet-V1. Ours significantly outperforms other methods in the pareto optimal curve. (b) Comparisons of the accuracy and latency trade-off among our framework, LeGR (Chin et al., 2020), LAMP (Lee et al., 2020), and unpruned MobileNet-V1. LAP raises the pareto curve of MobileNet-V1 compared to other approaches. (Inference time are measured on NVIDIA TITAN RTX GPU).

Table 4
Evaluation of each component in our framework under different overall sparsity for VGG-16, ResNet-50, and MobileNet-V1 on ImageNet. Reward$_{acc}$ and Reward$_{acc+lat}$ represent accuracy-oriented search protocol and latency-aware search protocol, respectively. The Top-1 accuracy and FLOP count of the pruned model are presented. M/B means million/billion ($10^6/10^9$), respectively.

Sparsity	Filter Selection	VGG-16		ResNet-50		MobileNet-V1	
		Reward$_{acc}$	Reward$_{acc+lat}$	Reward$_{acc}$	Reward$_{acc+lat}$	Reward$_{acc}$	Reward$_{acc+lat}$
25%	Magnitude	67.90%(8.26B)	67.88%(**7.79B**)	76.15%(2.23B)	76.13%(**2.11B**)	70.13%(315M)	70.12%(**302M**)
	ACS rank	**68.12%**(8.25B)	**68.11%(7.80B)**	**76.38%**(2.22B)	**76.36%(2.12B)**	**70.34%**(314M)	**70.34%(301M)**
50%	Magnitude	66.52%(3.86B)	66.51%(**3.65B**)	74.88%(1.07B)	74.85%(**1.01B**)	68.67%(144M)	68.65%(**138M**)
	ACS rank	**66.95%**(3.86B)	**66.93%(3.66B)**	**75.10%**(1.08B)	**75.08%(1.01B)**	**69.04%**(145M)	**69.02%(138M)**

strategy with accuracy-oriented policy and investigate the improvement brought by our dynamic-based importance evaluation method (i.e., ACS) and the magnitude-based filter selection criterion. From Table 4, It can be clearly observed that our latency-aware strategy significantly brings about more FLOPs reduction and ensures accuracy compared to the accuracy-oriented reward policy. For example, for MobileNet-V1 at 50% density, our strategy allows the model's FLOPs to be reduced from 144M to 136M under the magnitude-based filter selection, and from 145M to 138 under the ACS policy while maintaining the similar Top-1 accuracy. In addition, the filter selection guided by our ACS rank can lead to higher model accuracy than the magnitude-based policy, e.g., for MobileNet at 50% density, the Top-1 accuracy is improved by our selection method from 68.67% to 69.04% under the accuracy-oriented reward policy, and from 68.65% to 69.02% under our latency-aware strategy while the impact on the latency is negligible. Therefore, our proposed search strategy can find the more performance-friendly network structure, and our filter selection criterion can guide more accurate intra-layer filter pruning.

Different Filter Selection Criteria. There have been many prior works proposed to identify the important filters. Our filter selection method is compared with two recently emerged approaches (He et al., 2019; Lin et al., 2020) to show its effectiveness and advantages. A brief summary of them is as follows:

- **FPGM** (He et al., 2019). This criterion calculates the geometric median of each layer, and measures the distance between the geometric median and the filters as importance scores. Those with the smallest distance will be removed first.
- **HRank** (Lin et al., 2020). The low-rank feature maps contain less information, and thus pruning these part of feature maps produces very little damage to the model performance. Therefore, the average rank of feature maps is represented as the importance score of the filter that outputs them in this criterion.

In order to compare these different selection criteria, we evaluate their performance on VGG-19 with the dataset CIFAR-10. The network is trained from scratch with applying *uniform* pruning (the same sparsity for each layer) and fine-tuning under different overall sparsity (i.e., 10%, 20%, and 30%). The purpose of using these three pruning ratios is to gradually evaluate the effect of our filter selection method. Moreover, we further explore how much pruning rate this algorithm can achieve while producing model accuracy similar to other selection methods. Specifically, a total of 80 epochs are performed, 1st–69th epochs are performed for pruning, 70th–80th epochs are performed for fine-tuning, and finally the compact model with the highest accuracy is obtained. Each method is repeated 3 times, and the averaged result is presented. For fair comparison, all experimental settings are kept unchanged except for the filter selection method. In addition, the accuracy of the unpruned network trained from scratch is used as the baseline.

Fig. 6 shows the accuracy curves of different channel selection approaches. It can be clearly observed that our method enables the pruned model to obtain consistently and significantly higher accuracy compared with the other two methods under various sparsity. This demonstrates that it is reasonable and effective to employ the dynamic change of filters to represent their importance. Of note is that our method slightly improves the accuracy at 10% and 20% overall compression ratios compared to the baseline.

Fine-grained Dataset — Food-101. We also conduct experiments for VGG-19, ResNet-18, 34, 50, and 101 on Food-101. As

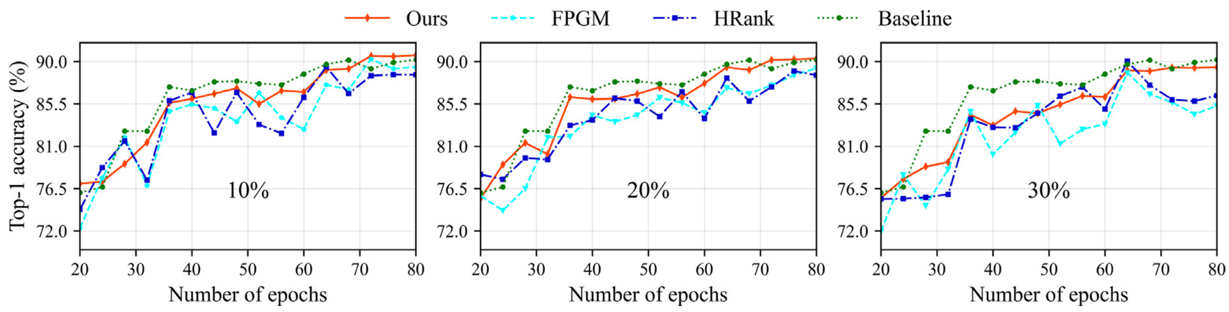


Fig. 6. Performance comparison of different filter selection methods: the VGG-19 model pruned on CIFAR-10 under different overall compression ratios (i.e., 10%, 20%, and 30%).

Table 5
Pruning results of VGG-19 and ResNet-18/34/50/101 on Food-101.

Network	Method	Top-1%	Top-1 acc. ↓(%)	Top-5%	Top-5 acc. ↓(%)	FLOPs ↓(%)	Parameters ↓(%)
VGG-19	VGG-19	73.95	–	91.36	–	–	–
	FPGM (30%) (He et al., 2019)	70.48	3.47	89.91	1.45	50.8	43.7
	HRank (30%) (Lin et al., 2020)	70.67	3.28	90.07	1.29	50.8	43.7
	Ours (10%)	73.09	0.86	90.93	0.43	18.9	25.2
	Ours (20%)	72.12	1.83	90.70	0.66	35.9	34.5
	Ours (30%)	**71.21**	**2.74**	**90.15**	**1.21**	50.8	43.7
	Ours (35%)	**70.73**	**3.22**	**90.11**	**1.25**	**58.2**	**48.3**
ResNet-18	ResNet-18	77.21	–	92.98	–	–	–
	HRank (30%) (Lin et al., 2020)	73.86	3.35	90.17	2.81	41.8	36.2
	Ours (30%)	**74.32**	**2.89**	**90.42**	**2.56**	41.8	36.2
ResNet-34	ResNet-34	78.41	–	93.21	–	–	–
	HRank (30%) (Lin et al., 2020)	76.05	2.36	93.03	1.33	41.1	36.5
	Ours (30%)	**76.31**	**2.10**	**93.12**	**1.24**	41.1	36.5
ResNet-50	ResNet-50	79.36	–	95.05	–	–	–
	FPGM (30%) (He et al., 2019)	77.07	2.29	93.62	1.43	41.8	41.6
	HRank (30%) (Lin et al., 2020)	77.04	2.32	93.04	2.01	41.8	41.6
	Ours (10%)	79.03	0.33	94.48	0.57	15.2	15.2
	Ours (20%)	78.14	1.22	93.95	1.10	28.6	29.0
	Ours (30%)	**77.35**	**2.01**	**93.91**	**1.14**	41.8	41.6
	Ours (34%)	**77.11**	**2.25**	93.56	1.49	**52.9**	**51.7**
ResNet-101	ResNet-101	80.03	–	95.42	–	–	–
	HRank (30%) (Lin et al., 2020)	78.96	1.07	93.36	2.06	42.2	41.9
	Ours (30%)	**79.38**	**0.65**	**94.25**	**1.17**	42.2	41.9

shown in Table 5, our selection method exceeds its counterparts in all aspects, including Top-1 and Top-5 accuracy, as well as FLOPs and parameters reduction. More specifically, for VGG-19 with the plain structure, our method removes 58.2% FLOPs and 48.3% parameters under 35% pruning ratio, while it only leads to 3.22% Top-1 and 1.25% Top-5 accuracy loss, obviously better than FPGM (He et al., 2019) and HRank (Lin et al., 2020). In addition, for ResNet with the residual block, our method also offers better results. For instance, 52.9% FLOPs and 51.7% parameters for ResNet-50 are reduced by our approach, only at the similar cost of accuracy drop to FPGM (He et al., 2019) and HRank (Lin et al., 2020) (2.25% Top-1 and 1.49% Top-5 losses), which surpasses these two state-of-the-art methods. Moreover, when comparing to HRank (Lin et al., 2020) for ResNet-101 at the 30% sparsity, higher accuracy are observed, i.e., 79.38% Top-1 vs. 78.96% Top-1 by HRank (Lin et al., 2020), 94.25% Top-5 vs. 93.36% Top-5 by HRank (Lin et al., 2020). Therefore, our method also achieves consistent superiority over other filter selection criteria on the fine-grained dataset.

ACS *vs* Adaptability. In this section, comparative experiments are performed to study the relationship between the ACS and the filter importance. Similar to the comparison experiments of different filter selection criteria, the VGG-19 is trained from scratch on CIFAR-10, and the filters with larger ACS are pruned at different compression ratios. As shown in Fig. 7(a), under 10% sparsity, the accuracy begins to decline from the 40th epoch and cannot be recovered. In addition, when setting more aggressive sparsity, the accuracy decreases earlier, i.e., the accuracy declines from 8th epoch at 20% and 30% sparsity ratios. This proves that the filters with larger changes have greater adaptability and are more important.

Cosine Similarity (CS) *vs* Adaptability. In this evaluation, we detect the relationship between the direction change of the filter and its adaptability. Specifically, first, we calculate the cosine similarity of the filter between different epochs. In this step, the adjusted operation is canceled, i.e., Eq. (4) and Eq. (5) in Section 3. Same as Eq. (6) in Section 3, using one minus the cosine similarity, which makes it be zero when the filter has no direction change, and the filter with larger change has larger value. Details are illustrated as follows:

$$\mathcal{S}^i \in \mathbb{R}^{O_i} = \left\{ \mathcal{C}^i_{j,k} = 1 - \frac{\mathcal{F}^i_{j,k-1}\mathcal{F}^i_{j,k}}{\|\mathcal{F}^i_{j,k-1}\| \, \|\mathcal{F}^i_{j,k}\|}, 1 \le j \le O_i \right\}, \tag{12}$$

where $\mathcal{C}^i_{j,k}$ is the cosine similarity of the jth filter in layer L_i between the $(k-1)$-th epoch and the kth epoch, $\mathcal{S}^i$ consists of the cosine similarity of all filters in layer L_i. $\mathcal{F}^i_{j,k}$ is the jth filter of layer L_i for kth epoch.

Second, the filters with smaller cosine similarity within layers are pruned under the same layer sparsity.

Fig. 7(b) shows that the model cannot achieve the accuracy obtained by our method. Specifically, at 10% sparsity, the model only achieves 87.52% top-1 accuracy (90.67% by ours). In addition, the model pruned at 20% and 30% sparsity even cannot converge.

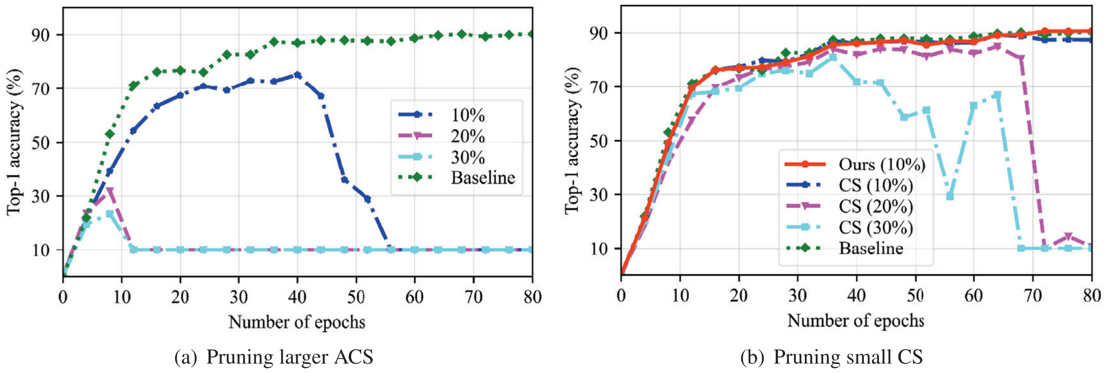


Fig. 7. Detecting the effect of different metrics of filter dynamic changes on filter selection. We train and prune VGG-19 from scratch on CIFAR-10 at different sparsity, i.e., 10%, 20%, and 30%. (a) The filters with larger ACS are selected to be pruned. (b) The filters with smaller cosine similarity (CS) are selected to be pruned. The Top-1 accuracy of VGG-19 trained without pruning is the baseline (green diamond).

Table 6
Comparisons of Filter Regrowth Strategy for MobileNet-V1 on ImageNet.

Network	Method	Top1-Acc	FLOPs
MobileNet (70.6%, 569M)	0.75 × base	68.40%	325M (57.11%)
	Mean-init	70.23%	303M (53.42%)
	Gradient-init	70.28%	302M (53.07%)
	Full-weight-init	70.26%	**301M (52.90%)**
	Zero-init	70.28%	**301M (52.90%)**
	Ours	**70.34%**	**301M (52.90%)**
	0.5 × base	63.70%	149M (26.18%)
	Mean-init	68.88%	139M (24.43%)
	Gradient-init	68.91%	140M (24.60%)
	Full-weight-init	68.94%	139M (24.43%)
	Zero-init	68.96%	**138M (24.25%)**
	Ours	**69.02%**	**138M (24.25%)**

The reason is that the cosine similarity cannot capture the overall changes of the filter (including magnitude and direction) like ACS. In addition, the filter cannot be compared with other filters in the same layer, e.g., no matter how great the difference between the magnitude changes of the two filters is, their changes in the same direction result in the same cosine similarity. This leads to the inability to accurately measure the adaptability of the filters and compare their importance. Therefore, the filters with larger adaptability may be pruned, and the remaining filters cannot make up for the representation ability of the pruned filters, leading to the model not being able to converge to the expected accuracy.

These two findings demonstrate that the heuristic algorithm we proposed is reasonable and effective for the filter selection, and can help us further understand CNNs.

Filter Regrowth Strategy. Previously many researchers adopted the strategy of initializing the pruned weight to zero during parameter exploration (Evci et al., 2020; He, Kang, et al., 2018; Mostafa & Wang, 2019; Zhuo et al., 2018). Experiments are conducted to evaluate the effectiveness of our filter regrowth strategy with MobileNet-V1 on ImageNet. Here all experimental settings are kept unchanged except for the filter regrowth strategy in our LAP framework. *Mean-init* means that the pruned filters are initialized to the mean of all filters in a layer. *Gradient-init* is to use the accumulative gradients of the pruned filters as the initial state of the next filter selection. *Full-weight-init* initializes the filter as the sum of its unpruned value and accumulative gradients. *Zero-init* initializes the regrowed filter to zero and continues to update it. As demonstrated in Table 6, it can be clearly seen that the effect of different regrowth strategies on the model latency is negligible, but they will significantly affect the model accuracy. Our method rewinds the pruned filters to their unpruned state, and outperforms other regrowth policies on model accuracy under different model size constraints.

5. Conclusion and discussion

In this paper, we present latency-aware automated pruning (LAP), an automated framework for filter pruning, which searches for the layer sparsity under arbitrary overall model size constraint with the accuracy and latency feedback in an end-to-end manner. Our framework can avoid suboptimal results of latency and labor-consuming pruning-selecting production process of compact high-performance models. For the compression policy within a layer, we also offer a novel filter selection algorithm, which captures the dynamic changes of filters as importance scores. Compelling results have been demonstrated for VGGNet, ResNet, and MobileNet on CIFAR-10, ImageNet, and Food-101. Our framework achieves consistent superiority compared with some state-of-the-art pruning methods under different networks. Our framework allows the inference speed of MobileNet-V1 to be improved from 125 ms to 76 ms without accuracy loss. LAP realizes the exploration of low-latency models while ensuring accuracy.

This research, however, is subject to several limitations. First, this algorithm can only prune pre-trained networks. This means that the network needs to be trained on a specific dataset in advance. The trained network is then pruned on the same dataset. This limits the application scope of the algorithm, i.e., it is difficult to use it to compress the untrained network. We plan to make improvements to the algorithm in the future, focusing on the determination process of layer sparsity, leading to that the algorithm can be applied to untrained networks. Second, this algorithm uses reinforcement learning to determine the layer sparsity, and the exploration process is somewhat time-consuming. We intend to design other exploration methods that can search reasonable layer sparsity more quickly or a method that can directly calculate layer sparsity, leading to that the algorithm can more efficiently compress dense networks.

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.

Acknowledgments

This work was partially supported by the National Key Research and Development Program of China (Nos. 2018YFB0204302, 2018YFB1701403), Programs of National Natural Science Foundation of China (Grant Nos. 62072165, U19A2058). This work was also sponsored by Zhijiang Lab, China (No. 2020KE0AB01).

References

Azarian, K., Bhalgat, Y., Lee, J., & Blankevoort, T. (2020). Learned threshold pruning. arXiv preprint arXiv:2003.00075.

Baker, B., Gupta, O., Naik, N., & Raskar, R. (2016). Designing neural network architectures using reinforcement learning. arXiv preprint arXiv:1611.02167.

Bossard, L., Guillaumin, M., & Van Gool, L. (2014). Food-101–mining discriminative components with random forests. In *European conference on computer vision* (pp. 446–461). Springer.

Chen, L.-C., Papandreou, G., Kokkinos, I., Murphy, K., & Yuille, A. L. (2014). Semantic image segmentation with deep convolutional nets and fully connected crfs. arXiv preprint arXiv:1412.7062.

Chin, T.-W., Ding, R., Zhang, C., & Marculescu, D. (2020). Towards efficient model compression via learned global ranking. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (pp. 1518–1528).

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., & Fei-Fei, L. (2009). Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition* (pp. 248–255). Ieee.

Denton, E. L., Zaremba, W., Bruna, J., LeCun, Y., & Fergus, R. (2014). Exploiting linear structure within convolutional networks for efficient evaluation. *Advances in Neural Information Processing Systems*, *27*, 1269–1277.

Dettmers, T., & Zettlemoyer, L. (2019). Sparse networks from scratch: Faster training without losing performance. arXiv preprint arXiv:1907.04840.

Ding, X., Ding, G., Guo, Y., Han, J., & Yan, C. (2019). Approximated oracle filter pruning for destructive cnn width optimization. arXiv preprint arXiv:1905.04748.

Evci, U., Gale, T., Menick, J., Castro, P. S., & Elsen, E. (2020). Rigging the lottery: Making all tickets winners. In *International conference on machine learning* (pp. 2943–2952). PMLR.

Girshick, R., Donahue, J., Darrell, T., & Malik, J. (2014). Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 580–587).

Han, S., Mao, H., & Dally, W. J. (2015). Deep compression: Compressing deep neural networks with pruning, trained quantization and Huffman coding. arXiv preprint arXiv:1510.00149.

He, Y., Kang, G., Dong, X., Fu, Y., & Yang, Y. (2018). Soft filter pruning for accelerating deep convolutional neural networks. arXiv preprint arXiv:1808.06866.

He, Y., Lin, J., Liu, Z., Wang, H., Li, L.-J., & Han, S. (2018). Amc: Automl for model compression and acceleration on mobile devices. In *Proceedings of the European conference on computer vision* (pp. 784–800).

He, Y., Liu, P., Wang, Z., Hu, Z., & Yang, Y. (2019). Filter pruning via geometric median for deep convolutional neural networks acceleration. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 4340–4349).

He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 770–778).

He, Y., Zhang, X., & Sun, J. (2017). Channel pruning for accelerating very deep neural networks. In *Proceedings of the IEEE international conference on computer vision* (pp. 1389–1397).

Hinton, G., Vinyals, O., & Dean, J. (2015). Distilling the knowledge in a neural network. arXiv preprint arXiv:1503.02531.

Howard, A. G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., et al. (2017). Mobilenets: Efficient convolutional neural networks for mobile vision applications. arXiv preprint arXiv:1704.04861.

Ioffe, S., & Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning* (pp. 448–456). PMLR.

Jaderberg, M., Vedaldi, A., & Zisserman, A. (2014). Speeding up convolutional neural networks with low rank expansions. arXiv preprint arXiv:1405.3866.

Krizhevsky, A., Hinton, G., et al. (2009). *Learning multiple layers of features from tiny images*. Citeseer.

LeCun, Y., Denker, J. S., & Solla, S. A. (1990). Optimal brain damage. In *Advances in neural information processing systems* (pp. 598–605).

Lee, J., Park, S., Mo, S., Ahn, S., & Shin, J. (2020). Layer-adaptive sparsity for the magnitude-based pruning. In *International conference on learning representations*.

Li, H., Kadav, A., Durdanovic, I., Samet, H., & Graf, H. P. (2016). Pruning filters for efficient convnets. arXiv preprint arXiv:1608.08710.

Li, G., Qian, C., Jiang, C., Lu, X., & Tang, K. (2018). Optimization based layer-wise magnitude-based pruning for DNN compression. In *IJCAI* (pp. 2383–2389).

Lillicrap, T. P., Hunt, J. J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., et al. (2015). Continuous control with deep reinforcement learning. arXiv preprint arXiv:1509.02971.

Lin, M., Ji, R., Wang, Y., Zhang, Y., Zhang, B., Tian, Y., et al. (2020). Hrank: Filter pruning using high-rank feature map. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (pp. 1529–1538).

Liu, Z., Mu, H., Zhang, X., Guo, Z., Yang, X., Cheng, K.-T., et al. (2019). Metapruning: Meta learning for automatic neural network channel pruning. In *Proceedings of the IEEE international conference on computer vision* (pp. 3296–3305).

Long, J., Shelhamer, E., & Darrell, T. (2015). Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 3431–3440).

Luo, J.-H., & Wu, J. (2020). Autopruner: An end-to-end trainable filter pruning method for efficient deep model inference. *Pattern Recognition*, Article 107461.

Luo, J.-H., Wu, J., & Lin, W. (2017). Thinet: A filter level pruning method for deep neural network compression. In *Proceedings of the IEEE international conference on computer vision* (pp. 5058–5066).

Morcos, A. S., Yu, H., Paganini, M., & Tian, Y. (2019). One ticket to win them all: Generalizing lottery ticket initializations across datasets and optimizers. arXiv preprint arXiv:1906.02773.

Mostafa, H., & Wang, X. (2019). Parameter efficient training of deep convolutional neural networks by dynamic sparse reparameterization. In *International conference on machine learning* (pp. 4646–4655). PMLR.

Paszke, A., Gross, S., Chintala, S., Chanan, G., Yang, E., DeVito, Z., et al. (2017). Automatic differentiation in pytorch.

Ren, S., He, K., Girshick, R., & Sun, J. (2015). Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems* (pp. 91–99).

Romero, A., Ballas, N., Kahou, S. E., Chassang, A., Gatta, C., & Bengio, Y. (2014). Fitnets: Hints for thin deep nets. arXiv preprint arXiv:1412.6550.

Shang, W., Sohn, K., Almeida, D., & Lee, H. (2016). Understanding and improving convolutional neural networks via concatenated rectified linear units. In *International conference on machine learning* (pp. 2217–2225).

Simonyan, K., & Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556.

Szegedy, C., Ioffe, S., Vanhoucke, V., & Alemi, A. (2016). Inception-v4, inception-resnet and the impact of residual connections on learning. arXiv preprint arXiv:1602.07261.

Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., et al. (2015). Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 1–9).

Tao, H., Li, J., Chen, Y., Stojanovic, V., & Yang, H. (2020). Robust point-to-point iterative learning control with trial-varying initial conditions. *IET Control Theory & Applications*, *14*(19), 3344–3350.

Tao, H., Li, X., Paszke, W., Stojanovic, V., & Yang, H. (2021). Robust PD-type iterative learning control for discrete systems with multiple time-delays subjected to polytopic uncertainty and restricted frequency-domain. *Multidimensional Systems and Signal Processing*, *32*(2), 671–692.

Wang, D., Zhou, L., Zhang, X., Bai, X., & Zhou, J. (2018). Exploring linear relationship in feature map subspace for convnets compression. arXiv preprint arXiv:1803.05729.

Wu, J., Leng, C., Wang, Y., Hu, Q., & Cheng, J. (2016). Quantized convolutional neural networks for mobile devices. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 4820–4828).

Xin, X., Tu, Y., Stojanovic, V., Wang, H., Shi, K., He, S., et al. (2022). Online reinforcement learning multiplayer non-zero sum games of continuous-time Markov jump linear systems. *Applied Mathematics and Computation*, *412*, Article 126537.

Xu, Z., Li, X., & Stojanovic, V. (2021). Exponential stability of nonlinear state-dependent delayed impulsive systems with applications. *Nonlinear Analysis. Hybrid Systems*, *42*, Article 101088.

Yu, J., & Huang, T. (2019). Autoslim: Towards one-shot architecture search for channel numbers. arXiv preprint arXiv:1903.11728.

Zhong, J., Ding, G., Guo, Y., Han, J., & Wang, B. (2018). Where to prune: Using LSTM to guide end-to-end pruning. In *IJCAI* (pp. 3205–3211).

Zhuo, H., Qian, X., Fu, Y., Yang, H., & Xue, X. (2018). Scsp: Spectral clustering filter pruning with soft self-adaption manners. arXiv preprint arXiv:1806.05320.