



A scalable channel pruning framework with computation-adaptive artificial bee colony

Yijie Chen ^{a,1}, Chengyang Sun ^a, Yaqi Yao ^{b,*,2}, Rui Li ^{b,*,2}, Renfa Li ^{b,2}, Keqin Li ^c

^a Northwest A&F University, Yangling, 712100, Shaanxi, China

^b Hunan University, Changsha, 410082, Hunan, China

^c State University of New York, NY, 10003, New Paltz, USA

ARTICLE INFO

Keywords:

Channel pruning
Automatic structure search
Structure mapping
Artificial bee colony

ABSTRACT

Channel pruning, known for its hardware-friendly characteristics of structural pruning, has been widely adopted to slim deep neural networks. However, the increasingly complex coupled structures in neural networks pose new challenges to channel pruning methods. First, most existing channel pruning approaches exhibit customized characteristics, making them difficult to adapt to the multiple new neural networks derived from coupled structures embedded in neural networks. Second, the intricate inter-layer dependencies within coupled structures are crucial for enhancing model performance, requiring careful pruning to prevent severe performance degradation in the pruned model. Meanwhile, continuously evolving coupled structures further exacerbate these challenges. To achieve high-ratio pruning of deep neural networks while preserving model performance and ensuring the scalability of the pruning method, this paper proposes a computation-adaptive artificial bee colony approach for scalable channel pruning. First, this paper proposes a structure mapping strategy to construct a scalable search space, which decouples complex networks by establishing different mapping relationships between network architectures and search structures. Then, this paper designs a computation-adaptive artificial bee colony algorithm that reduces fitness estimation bias by utilizing local evaluation consistency among sampling networks. Finally, a cell-level search space construction method is introduced to exponentially reduce the search space. Experimental results demonstrate the superiority of the proposed method on computer vision tasks. Additionally, ablation studies and visualizations verify the effectiveness of the proposed approach.

1. Introduction

Channel pruning has emerged as a highly effective technology to accelerate the inference speed of deep neural networks (DNNs) in engineering applications [1]. It removes redundant channels from a DNN to meet the requirements of limited model parameters and computation in a resource-constrained environment. Existing channel pruning methods can be divided into channel importance based pruning methods [2] and structure importance based pruning methods [3]. The former removes unimportant channels through channel importance discrimination, while the latter converts the pruning problem into a structure search problem, selecting important structural components

to identify lightweight subnets. These methods can effectively prune classic neural network models such as MobileNet and ResNet, but they require specialized design for each model, making it difficult to quickly extend the pruning methods to new neural network models.

The outstanding performance of deep neural networks in vision and language tasks has driven progress across various industries. To better apply DNNs, researchers study different network components [4] and continuously iterate new versions of neural networks. Neural network components consist of various fundamental modules—such as the convolutional layer (CONV), the batch normalization layer (BN), the activation functions—combined in different network topologies and integrated into classic neural networks to enhance model performance.

* Corresponding authors.

E-mail addresses: yijiechen@nwfufu.edu.cn (Y. Chen), 2022013377@nwfufu.edu.cn (C. Sun), yaqiyao@hnu.edu.cn (Y. Yao), rui@hnu.edu.cn (R. Li), lirenfa@hnu.edu.cn (R. Li), lik@newpaltz.edu (K. Li).

¹ Yijie Chen is a lecturer at College of Information Engineering, Northwest A&F University, Yangling, Shaanxi, China. Chengyang Sun is pursuing his bachelor's degree at College of Information Engineering, Northwest A&F University.

² Yaqi Yao, Rui Li, and Renfa Li are with the Key Laboratory for Embedded and Network Computing of Hunan Province, the Hunan Engineering Research Center of Advanced Embedded Computing and Intelligent Medical Systems, College of Computer Science and Electronic Engineering, Hunan University, Changsha, China

<https://doi.org/10.1016/j.swevo.2026.102563>

Received 18 January 2026; Received in revised form 7 September 2026; Accepted 2 October 2026

Available online 15 October 2026

2210-6502/© 2026 Elsevier B.V. All rights reserved, including those for text and data mining, AI training, and similar technologies.

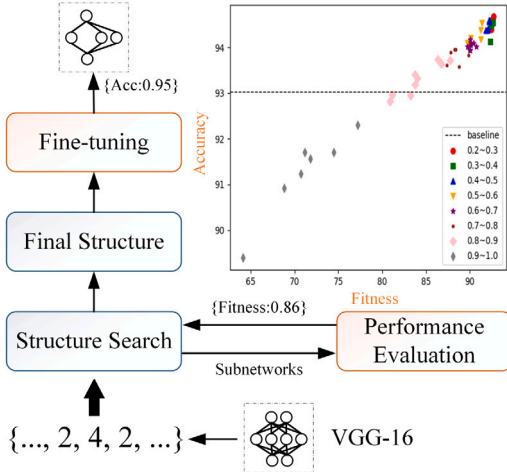


Fig. 1. The model correlation between the fitness and the final accuracy of pruned models. All models are marked different colors according to their FLOPs reduction. The fitness is the subnetwork's score in the search process. The final accuracy is subnetwork's performance after fine-tuning.

The intricate dependencies within these modules are often highly sensitive to the performance of neural network models, requiring careful handling during channel pruning to prevent model degradation. In order to prune the coupling structures of DNNs with varying complexity, Fang et al. [5] proposed DepGraph, which constructs a dependency graph for a DNN and groups the coupled structures into distinct sets. Then it applies sparsity training to emphasize channel importance within these coupled structures, calculating channel importance using the L_2 -norm of the grouped parameters. These methods prune based on local parameter magnitude comparisons, and their multi-stage nature hinders scalability to other models.

Compared to channel importance-based pruning methods, structure importance based pruning methods prune DNN automatically, which search for a lightweight structure from the search space constructed by the original model. ABCPruner [6] adopts the artificial bee colony (ABC) algorithm to search only for plain structures. DAIS [7] proposes a differentiable search of annealing indicators to prune simple coupling structures, which sets the channel indicator to prune simple coupling structures in ResNet and MobileNet. This type of channel pruning method addresses the pruning of model parameters at the channel level from a structural perspective, offering inherent scalability advantages for continuously evolving model architectures. However, existing methods fail to address the scalability issue in channel pruning and only handle localized coupled structures. The underlying reason is that current approaches lack a unified representation for diverse structures, making it difficult for them to prune complex coupled relationships from a global perspective.

Besides, channel pruning methods based on architecture search suffer from fitness estimation bias, leading to suboptimal search results. This manifests as a weak correlation between the importance scores (termed fitness) of substructures and their ultimate performance. To illustrate this issue, we randomly sampled several architectures from the search space of VGG-16 and recorded the floating-point operations (FLOPs) range for each architecture. As shown in Fig. 1, the correlation between fitness and final performance is chaotic for the sampled structures, which results in inefficient architecture search. Fortunately, as can be observed from the figure, the fitness of sampled structures with similar FLOPs tends to cluster together. This phenomenon suggests that for the sampled subnetworks, both fitness and final performance may be influenced by the model's scale (i.e., parameter count and computational cost). This phenomenon describes the absence of the

local evaluation consistency during search process: for two subnetworks with identical fitness scores, the larger model is more likely to achieve higher final performance due to its expanded parameter space. Given that computational cost is a critical metric in channel pruning, this paper incorporates FLOPs as a constraint in the architecture search process.

To sum up, to achieve high-ratio pruning of neural networks while maintaining their performance, this paper proposes a scalable computation-adaptive structure search framework. First, a structure mapping method is proposed to achieve the scalability among different DNNs. Then, a novel computation-adaptive Artificial Bee Colony (FC-ABC) algorithm is presented to search for the optimal pruned structure. Finally, a search space reduction method is proposed to minimize the resources required for the search process. The contributions of this work can be summarized as follows:

- A structure mapping method is proposed to flexibly define the search space, which enables mutual conversion between the model structure and the search structure by establishing different mapping relationships.
- A novel computation-adaptive artificial bee colony algorithm is designed to utilize local evaluation consistency and reduce fitness estimation bias by incorporating FLOPs constraints.
- A cell-level search space method is proposed to control the pruning granularity and reduce the search space, effectively addressing the issue of potential layer over-pruning during channel pruning by defining cell size.

2. Related work

2.1. Importance-based pruning

The channel importance based pruning methods remove relatively unimportant channels by comparing the channels' importance in a layer-level way or global way. The metrics for evaluating the channel importance include weight norm [8], L_p regularization [9], Taylor expansion of cross-entropy [10], the difference of final response layer [11], feature-maps relevant information [12], KL-divergence [13], kernel similarity [2]. Typically, these methods involve at least three stages: importance enhancement, model pruning, and fine-tuning. To prune the increasingly complex coupled structures, Fang et al. [5] presented a group-level pruning approach, dubbed DepGraph, whose process consists of model grouping, sparsity training, model pruning, and fine-tuning. The complexity of the process poses significant challenges to prune a new deep neural network. Moreover, their reliance on local importance signals often overlooks global topological dependencies, leading to substructures that are efficient in isolation but suboptimal when considering the holistic architectural integrity required for effective fine-tuning.

2.2. Automatic structure search

The automatic structure search applies neural architecture search [14] to the field of channel pruning. Structure search typically consists of three main components: the definition of the search space, the structure search process itself, and the performance evaluation strategy for subnetworks. A subnetwork's structural definition is derived from a combination of various channel numbers [15] or channel indicators [7]. These methods essentially use 0-1 vectors to construct channel-wise pruning masks for arbitrary convolutional layers, where 0 indicates removal of the corresponding channel and 1 indicates retention. To obtain a good pruning model, Guo et al. [16] presented differentiable Markov channel pruning to search the optimal sub-structure. Gao et al. [3] proposed a bi-level optimization-based model for channel pruning, which searches for the optimal pruned structure by leveraging short-term gradient changes. Hu et al. [17] conducted a training

framework (BNSC) to slim DNN automatically based on online channel pruning. Lu et al. [18] presented a strategy of Growing-before-Pruning (Gbp) to explore space within architecture search for pruning DNNs. Existing methods search for the optimal solution in the search space in a global search manner during the search process, neglecting the fitness estimation bias caused by the computational complexity scale of sub-networks. This often leads to the optimal solution not satisfying the target computational complexity constraint.

2.3. Evolutionary computation

Evolutionary computation (EC) has been extensively employed to solve complex optimization problems in deep learning, where its population-based search mechanism enables efficient exploration of high-dimensional, discrete, and non-differentiable spaces. This property aligns well with the challenge of locating effective substructures within a large and irregular pruning search space. Consequently, within a defined search space for DNNs, numerous EC-based methods have been developed to explore optimal model structures, employing algorithms such as genetic algorithms, particle swarm optimization, and differential evolution. Zhao et al. introduced differential evolution and gradient descent strategies into an improved Hummingbird algorithm to counter its susceptibility to local optima [19]. Tang et al. developed an evolutionary NAS method that replaced uninformed perturbations with distribution-aware structural variation to improve search efficiency [20]. Tong et al. designed a G-EvoNAS, which adopts a gradually searching mechanism to search diverse network architectures with different depths and uses evolutionary algorithm to optimize the module structure at each stage [21]. Existing studies have devoted efforts to enhancing evolutionary computation algorithms for the efficient handling of specific tasks. Focus on the artificial bee colony, Ma et al. proposed a diversified initialization mechanism within an reinforcement learning-integrated discrete artificial bee colony (RLDABC) algorithm for solving large-scale instances [22]. Lu et al. presented HFABC to mitigate poor initial population quality and inefficient search equations of ABC. For that, HFABC employs a population initialization strategy and a dynamic divide-and-conquer search strategy [23]. The above methods improve the ABC algorithm through improved initialization and search strategies.

In conclusion, to effectively prune DNNs, this paper builds flexible representation for coupled structures in DNNs to implement scalable pruning. And then this paper mitigates poor initial population quality and inefficient search process of ABC by controlling the FLOPs of subnetworks during the search stage.

3. Scalable channel pruning via computation-adaptive structure search

This paper presents a scalable channel pruning framework, whose workflow is illustrated in Fig. 2. Firstly, structure mapping is presented to build search space for DNNs. Then, computation-adaptive artificial bee colony is proposed to search for a suitable structure, whose process includes initialization, searching, and updating. After the search is completed, the best structure in the update set will be mapped back to a pruned network for further fine-tune. To shrink the search space, a cell-level search space method is proposed in the mapping process.

3.1. Problem statement and formulation

This section formulates the channel pruning problem based on structure search. For any pre-trained model N , its structure is denoted as $B = (b_1, b_2, \dots, b_l, \dots, b_L)$, where b_l is the channel number of the l th CONV layer and L is the number of CONV layer. Channel pruning aims to decrease any b_l while maintaining the model performance. This paper treats the channel pruning problem as a structure searching problem to search for an optimal pruned structure. Given the training

set $\mathcal{T}_{train}$ and the testing set $\mathcal{T}_{test}$, the process of structure search consists of sampling subnetworks from a DNN, iteratively evaluate and select the subnetworks, and output the subnetwork with the best evaluation score (fitness). Let $\mathcal{N}'$ represent any pruned model, with its model structure denoted as $B' = (b'_1, b'_2, \dots, b'_l, \dots, b'_L)$, where $b'_l \leq b_l$ represents the remaining channel number in the l th CONV layer. The structure search problem can be formulated as

$$(B')^* = \arg \max_{B'} acc(\mathcal{N}'(B', W'; \mathcal{T}_{train}); \mathcal{T}_{test}), \quad (1)$$

where W' is the weights of B' , which is preserved from the weight of N . And $acc(\cdot)$ denotes the accuracy on $\mathcal{T}_{test}$.

In initialization process, the model structure B can be adopted as the search space $(b_1, b_2, \dots, b_l, \dots, b_L)$. However, in a complex DNN, dependencies may exist between any two CONV layers, which are manifested by the requirement that the number of output channels must remain consistent to maintain the network's functionality. It can be dubbed as couple structure. For any DNN with couple structures, the structure search problem is formulated as

$$(B')^* = \arg \max_{B'} acc(\mathcal{N}'(B', W'; \mathcal{T}_{train}); \mathcal{T}_{test}) \quad (2)$$

$$\text{s.t. } \exists i \neq j, b'_i = b'_j.$$

where $b'_i = b'_j$ represents that the channel number of i th CONV layer must be consistent with the channel number of j th CONV layer during the pruning process. Therefore, the coupled structures bring a difficult problem for structure search. To solve it, this paper proposes structural mapping to simply the coupled dependencies.

3.2. Structure mapping

Structural mapping groups coupled layers and maps them to the same search vector. Depending on whether the number of channels needs to be consistent between the two convolutional layers, DNNs can be divided into two categories: plain structures (Fig. 3) and coupled structures (Fig. 4). Structure mapping eliminates the constraint in Eq. (2) and builds a general search space for both plain and coupled structures in a DNN. For example, the inter-layer dependency in a DNN can be formulated as a dependency graph $DG = [(0, 0), (0, 3), (2, 5), (0, 0), (3, 0), \dots]$, where the index represents layer index, the values greater than 0 represent inter-layer dependencies, and values of 0 indicates no inter-layer dependencies. After structure mapping, the model architecture $B = (b_1, b_2, \dots, b_L)$ is mapped to the search structure $B^1 = (b^1_1, b^1_2, \dots, b^1_{L^1})$, where $L^1 \leq L$ represents the length of the search structure.

In plain networks, there are straightforward dependencies only between two adjacent layers; pruning such a CONV layer only requires considering the change of its input and output. Therefore, for **plain networks**, as there are no intricate connections between their CONV layers, a one-to-one mapping between the model architecture and the search structure is established. To illustrate this, the key parts from VG-Net [24] and GoogLeNet [25] are selected. For example, consider the simple plain network shown in Fig. 3 (1), VGNet's dependency graph $DG_{vgg} = [(0, 0), (0, 0), (0, 0), \dots]$ and its architecture can be mapped as $B^1\text{-vgg} = (in, x1, x2, x3, \dots)$. Similarly, for the complex plain network depicted in Fig. 3 (2), the main structure of GoogLeNet's architecture (Inception) can be mapped as $B^1\text{-inet} = (in, x1, x2, x3, x4, x5, x6, x7, \dots)$. As there are no complex dependencies among their model architecture, a one-to-one mapping relationship is feasible.

In contrast, multiple CONV layers in coupled networks have intricate connections, which give rise to complex dependencies. For instance, in residual blocks, one CONV layer depends on the front two CONV layers (Fig. 4 (1)). Among them, the number of output channels for both CONV layers needs to keep consistent during channel pruning. Therefore, for coupled networks, multiple coupled CONV layers can be regarded as a single unit in the search structure by adopts a many-to-one mapping relationship between model architecture and search

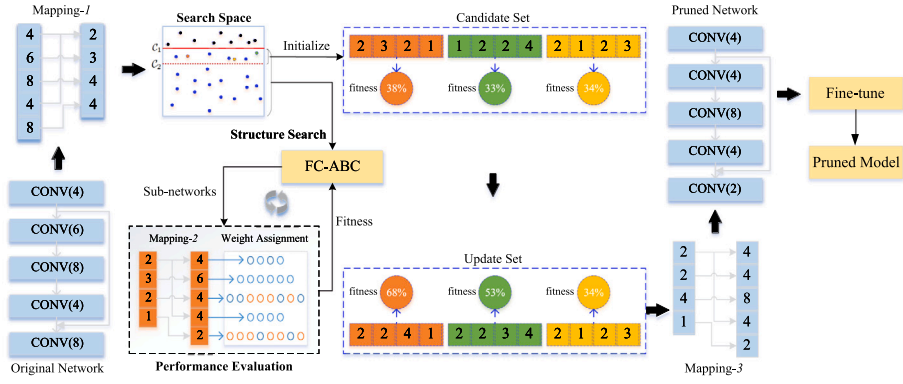


Fig. 2. The framework of automatic structure search under FLOPs constraint (C_1, C_2) using 2-level search space. Mapping- x means that the mapping process between a model architecture with the corresponding search structure.

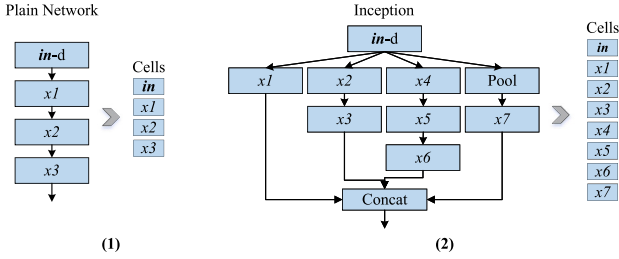


Fig. 3. Structure mapping for plain networks between their model architecture with search structure.

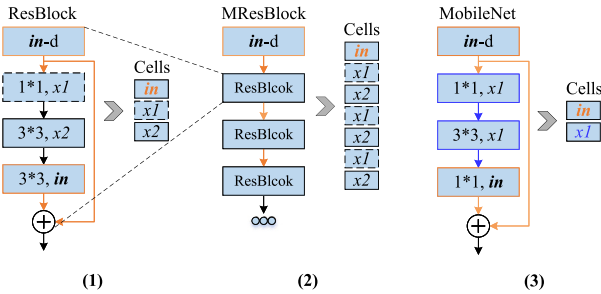


Fig. 4. Three mapping relationships for coupled network between the model architecture and search structures. (1) is two types of residual blocks. (2) is two types of multiple residual block connected in series. (3) is the main structure of MobileNet.

structure. The key parts from ResNet [26] and MobileNet [27] are selected as simple coupled structures. In Fig. 4, the CONV layer with dotted rectangle can be removed, and the highlighted directed lines indicate the coupled relationship between the outputs of multiple CONV layers. Through treating coupled layers as a single unit, the mapping relation of $B^1\text{-res1} = (in, x2, \dots)$ and $B^1\text{-res2} = (in, x1, x2, \dots)$ for two types of residual blocks of the ResNet are conducted, respectively. Moreover, for the essential structure in MobileNet with two different coupled structure (Fig. 4 (3)), its dependency graph is $DG_{mnet} = [(0, 4), (0, 3), (2, 0), (1, 0), \dots]$ and the model structure can be mapped as $B^1\text{-MNet} = (in, x1, \dots)$ based on the DG_{mnet} , which clearly demonstrates a many-to-one mapping relationship. Moreover, for the complex coupled structure as multiple coupled modules connected in series, whose coupled layers are also treated as a single unit in the search structure. In Fig. 4 (2), the MRResBlock structure includes multiple residual blocks in series, wherein all ResBlocks' output are coupled. For that, the mapping relationship can be established as $B^1\text{-Mres1} = (in, x2, x2, x2, \dots)$ and $B^1\text{-Mres2} = (in, x1, x2, x1, x2, x1, x2, \dots)$, respectively. It is noted that

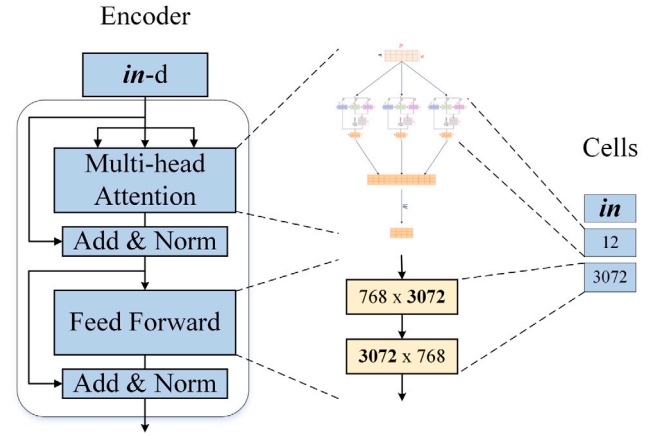


Fig. 5. Structure mapping over the encoder module.

for the plain structures within coupled networks, a one-to-one mapping relationship is enough.

This paper aims to achieve structured pruning of deep neural networks, which hinges on how to better group the complex coupled components within DNNs. To this end, we decompose DNNs into their constituent components. Recently, with the growing popularity of Transformer-series models, our method has also been extended to ViT models. Taking ViT-base as an example, besides plain networks, its most critical components are the MSA module and the MLP module. Inspired by existing work, we treat the attention heads in MSA as basic units and search for the header count, while simultaneously taking the channel dimension between the two Linear layers in the MLP as another search objective (as shown in Fig. 5). Moreover, our method can be extended to ConvNeXt and Swin Transformer models, due to the core components of ConvNeXt primarily include residual connections and depthwise separable convolutions, while Swin Transformer still centers on MSA and MLP modules. Since our method has already addressed the adaptation of such fundamental components, it is applicable to both model types in principle.

Structure mapping is utilized in three stages within the proposed framework. The dependency graph DG of a DNN maintains mapping consistency across networks with varying coupling patterns. As shown in Fig. 2, before the initialization process, the model architecture is mapped to the search structure (Mapping-1). During the search process, the sampled subnetwork structures need to be mapped back to the original model architectures (Mapping-2). After the searching process is concluded, the unique search structure of the optimal subnetwork that is obtained needs to be mapped back to the model architecture

(Mapping-3). The Mapping-2 and Mapping-3 perform similar work, mapping the obtained search structure to a new model architecture. The search structure is mapped back to a model structure composed of channel arrays for each layer. The corresponding channel parameters are selected from the original model parameters for parameter initialization. Filter assignment is adopted, where all channels in a convolutional layer are sorted by their $L1$ -norm, and the top-ranked channel parameters are selected according to the new architecture.

Based on structure mapping, the network structure $B = (b_1, b_2, \dots, b_L)$ is transformed into a search structure $B^1 = (b^1_1, b^1_2, \dots, b^1_{L^1})$, where $L^1 \leq L$ represents the length of the search structure. The original network structure pruning problem is equivalent to finding a suitable search structure within the search space: randomly selecting any sub-network from B^1 , which retains L^1 layers but has different random channel numbers in various layers. The number of potential combinations that can be obtained from the search structure B^1 is $\prod_{i=1}^{L^1} b^1_i$, which can be extremely large. Moreover, the number of combinations grows by a factor of b^1_i as the number of layers increases, where $b^1_i \in \{16, 32, \dots, 1024, 2048, \dots\}$. Searching such a vast search space is computationally expensive, and exhaustive search using a computer is clearly infeasible. Additionally, to reduce manual intervention during the search process, intelligent optimization algorithms are required for the automatic search of structures.

3.3. Computation-adaptive artificial bee colony

The ABCPruner [6] only searches for plain structures within the DNN and fails to consider the impact of local evaluation consistency on fitness estimation. Different with ABCPruner, this paper presents the novel computation-adaptive artificial bee colony algorithm FC-ABC. FC-ABC optimizes search efficiency by pre-setting a search starting point, recording any searched structure, and shrinking the search space. Then this paper dynamic control the FLOPs of sub-networks to mitigate the intrinsic fitness estimation bias (as described in Fig. 1). The structure search problem (Eq. (1)) can be further formulated as follows:

$$(B^1)^* = \arg \max_{B^1} acc(\mathcal{N}'(B^1, W'; \mathcal{T}_{train}); \mathcal{T}_{test}), \quad (3)$$

s.t. $FLOPs \leq C_1$

where C_1 represents the upper limit of the FLOPs, which is obtained by pruning ratio pr and is used to control the maximum FLOPs of any search structure during the search process. FC-ABC aims to find a suitable structure within the search space, which consists of initialization stage and searching stage.

Algorithm 1 summarize the initialization process. Firstly, we initialize a candidate set S with n structures, denoted as $\{B_j^{1'}\}_{j=1}^n$. To reduce the fitness evaluation bias, FC-ABC constrains the initialization of individuals within the population by presetting the search starting point (the FLOPs of candidate structures) according to two FLOPs thresholds $\{C_1, C_2\}$. Each structure must obey the FLOPs constraint as $flops_{(B_j^{1'})} \in [C_1, C_2]$. C_1 is applied to the phases of initialization and searching, which is the upper bound of the computation for adjusting the FLOPs of search structures. C_2 is used to limit the lower bound of the computation of any initial candidate structure, which is applied only to the initialization phase. Given a pruning rate of FLOPs pr , the FLOPs $flop_{\mathcal{N}'}$ of $\mathcal{N}'$ is fixed. The FLOPs thresholds are calculated by

$$\begin{cases} C_1 = (1 - pr) \times flop_{\mathcal{N}'} \\ C_2 = (1 - pr - constant) \times flop_{\mathcal{N}'} \end{cases}, \quad (4)$$

where $constant \in [0.01, 0.1]$ is an empirical value. The values (1% and 10% of the total FLOPs of the pre-trained model) are empirically derived from the architectural characteristics of the target network.

For generating a structure $B_j^{1'} = (x_1, x_2, \dots, x_{L^1})$, its i th element x_i is randomly sampled from $\{1, 2, \dots, b^1_i\}$. To ensure that each candidate structure $B_j^{1'}$ obeys $flops_{(B_j^{1'})} \in [C_1, C_2]$ constraint, FC-ABC adaptively adjusts the generated structures during the search process: when the

Algorithm 1 Initialization

Input: The number of candidates: n , Pruning Ratio: pr , FLOPs constraints: $\{C_1, C_2\}$, Searched Set : $\mathcal{G}$, Candidate Set: S

Output: $S, \mathcal{G}$

```

1:  $j \leftarrow 1$ 
2: while  $j < n$  do
3:   Random sampling generates candidate  $B_j^{1'}$ .
4:   while True do
5:     Calculate the FLOPs  $flops_{(B_j^{1'})}$  of  $B_j^{1'}$ .
6:     if  $flops_{(B_j^{1'})} > C_1$  then
7:       Decrease the values in  $B_j^{1'}$ .
8:     else if  $flops_{(B_j^{1'})} \leq C_2$  then
9:       Increase the values in  $B_j^{1'}$ .
10:    else
11:      break.
12:    end if
13:  end while
14:  if  $B_j^{1'} \notin \mathcal{G}$  then
15:    Calculate the fitness of  $B_j^{1'}$ .
16:    Add  $B_j^{1'}$  into  $S, \mathcal{G}$ .
17:     $j \leftarrow j + 1$ 
18:  end if
19: end while

```

$flops_{(B_j^{1'})}$ value is greater than C_1 , a number of values (default is 3) are randomly selected from $B_j^{1'}$ and decreased sequentially. When the $flops_{(B_j^{1'})}$ value is less than C_2 , similarly, a number of values (default is 3) are randomly selected from $B_j^{1'}$ and increased sequentially. This process is repeated iteratively until the $flops_{(B_j^{1'})}$ satisfies the $[C_1, C_2]$ constraint. Then this structure is mapped back to its model structures to obtain its fitness and it will be added to a searched structure set $\mathcal{G}$, that is, structural records. The structures in $\mathcal{G}$ will not be searched again, improving the search efficiency. The above sampling process is repeated until n structures are obtained.

After initialization is completed, the search process (Algorithm 2) can start. Three types of bees, employed bee, onlooker bee, and scout bee are released sequentially in one iteration cycle. More details are summarized as follows.

Employed Bee (Line 3-Line 17 in Alg. 2). The employed bee generates a new structure G'_j for each candidate structure $B_j^{1'}$. The i th element of G'_j is defined as below:

$$G'_{ji} = [b_{ji}^{1'} + r \cdot (b_{ji}^{1'} - b_{gi}^{1'})]$$

$$r \in \begin{cases} [-0.5, 0.0], & flops_{(G'_j)} > C_1 \\ [-0.5, 0.5], & flops_{(G'_j)} \leq C_1 \end{cases}, \quad (5)$$

where $g \neq j$ represents the i th element of the g th candidate structure and the i th element of the j th candidate structure in S , respectively. And r is a random number used to control the direction of FLOPs variation of a search structure during the search process. The generated G'_j must satisfy: (1) it has not been searched; (2) its computational cost is less than C_1 . If its FLOPs is greater than C_1 , FC-ABC will regenerate a new G'_j according to Eq. (5). Until G'_j satisfies the above two constraints, then it will be mapped back to its model structure to get its fitness. Subsequently, based on the fitness of G'_j and $B_j^{1'}$, it is determined whether G'_j should replace $B_j^{1'}$.

Onlooker Bee (Line 18-Line 24 in Alg. 2). The onlooker bee further updates the set of candidate structure S . Randomly, a search structure in the set is chosen with a probability related to its fitness, the probability is given by:

$$P_j = 0.9 \cdot \frac{fit_{B_j^{1'}}}{\max(\{fit_{B_j^{1'}}\}_{j=1}^n)} + 0.1. \quad (6)$$

Algorithm 2 FC-ABC

Input: Cycles: $\mathcal{T}$, FLOPs constraint: C_1 , Pruning Ratio: pr , Searched set: $\mathcal{G}$, Candidate Set: S , Max update times: $\mathcal{M}$, Candidates number: n
Output: $(B^{l'})^*$

```

1: Initialize  $S$  via Algorithm 1
2: for  $h = 1 \rightarrow \mathcal{T}$  do
3:   for  $j = 1 \rightarrow n$  do
4:     while True do
5:       Generate candidate  $G'_j$  via Eq. (5).
6:       Calculate the FLOPs  $flops_{(G'_j)}$  of  $G'_j$ .
7:       if  $flops_{(G'_j)} < C_1$  and  $G'_j \notin \mathcal{G}$  then
8:         break;
9:       end if
10:    end while
11:    Calculate the fitness of  $G'_j$  and add it into  $\mathcal{G}$ .
12:    if  $fit_{B^{l'}_j} < fit_{G'_j}$  then
13:       $B^{l'}_j = G'_j$ ;  $fit_{B^{l'}_j} = fit_{G'_j}$ ;  $t_j = 0$ .
14:    else
15:       $t_j = t_j + 1$ .
16:    end if
17:  end for
18:  for  $j = 1 \rightarrow n$  do
19:    Calculate the probability of  $P_j$  via Eq. (6).
20:    Generate a random number  $\sigma_j \in [0, 1]$ ;
21:    if  $\sigma_j \leq P_j$  then
22:      Repeat: Line 4-Line 16.
23:    end if
24:  end for
25:  for  $j = 1 \rightarrow n$  do
26:    if  $t_j > \mathcal{M}$  then
27:      while True do
28:        Generate candidate  $G'_j$  via Eq. (5).
29:        Calculate the FLOPs  $flops_{(G'_j)}$  of  $G'_j$ .
30:        if  $flops_{(G'_j)} < C_1$  and  $G'_j \notin \mathcal{G}$  then
31:          break.
32:        else
33:          Decrease the value of  $G'_j$ .
34:        end if
35:      end while
36:      Repeat: Line 12-Line 16.
37:    end if
38:  end for
39: end for
40: Output the optimal structure  $(G')^*$ .
41: Prune model and fine-tune the pruned model.

```

The update process of the chosen structure is the same as that in employed bee (Line 4-Line 16). For $B^{l'}$, the better its fitness, the higher the probability of it being selected, making it easier to be updated following the rule of Eq. (5). Through this iterative process, the set of candidate structures will be progressively updated, ultimately finding the optimal search structure.

Scout Bee (Line 24-Line 38 in Alg. 2). If a candidate structure $B^{l'}$ has not been updated after exceeding $\mathcal{M}$ (default:5) iterations, the scout bee will reinitialize it to generate a new candidate structure. Similar to the initialization phase, FC-ABC adaptively adjusts the values in the generated structures to meet the computational constraints. In this phase, the generated structures are only required to have their FLOPs below the C_1 value, which enables FC-ABC to explore lighter network architectures.

Fitness Calculation. Fitness indicates the structure importance for selecting better candidate structures in FC-ABC. Each search structure

is mapped back to the model architecture to get its fitness, which is given by

$$fit_{(B^{l'})} = acc(\mathcal{N}'_j(B^{l'}_j, W'_j; \mathcal{T}_{train}); \mathcal{T}_{test}), \quad (7)$$

where W'_j represents the weights retained from the original weights W of the pre-trained model $\mathcal{N}$. Inspired by [28], for a new CONV layer with remaining k channels, we retain the top- k channels' weights from the original CONV layer during the process of weight assignment. The rank is derived from sorting the $L1$ norms of all channels in the CONV layer. For each CONV layer, the $L1$ norms are sorted in descending order. After weight assignment is completed, any sampled model $\mathcal{N}'_j$ is trained $\mathcal{E}$ (default:2) epochs on $\mathcal{T}_{train}$. Then the average of the accuracy tested on $\mathcal{T}_{test}$ is obtained as its fitness.

During the whole process, a value in the search structure may fall outside the defined range of the search space. When this value is less than 0, set it to 1, which is also the minimum pruning unit (*cell* mentioned in Section 3.4); when the value exceeds the defined maximum, set it to the specified maximum value. To enhance search efficiency, the proposed algorithm treats candidate structures that have undergone fitness calculation as compliant with computational constraints and marked for search. All searched subnetwork structures are recorded in the set $\mathcal{G}$ of searched structures. During the search process, only structures that have not been searched before are explored. After the entire search phase, an optimal structure $(B^{l'})^*$ is selected from the candidate set S . Then $(B^{l'})^*$ is mapped back to the model structure and subjected to further fine-tuning.

3.4. Cell-level search space

This section defines a cell-level search space to scale search space. For search structure B^1 , its search space is the number of potential combinations. The sizes of the DNN's search space can be calculated by $\prod_{i=1}^{L^1} b^1_i$. Moreover, the number of combinations grows by a factor of b^1_i as the number of layers increases, where $b^1_i \in \{16, 32, \dots, 1024, 2048, \dots\}$. Searching for the optimal structure in such a vast search space is computationally expensive, and exhaustive search is clearly infeasible in a computer. To reduce the size of the search space while preserving its integrity, we adopt a cell-level search space to make a rule that each cell in the search space consists of one or more channels.

Whether in B or B^1 , 1 represents one channel. Let a basic unit contain *cell* channels, whose value is a real number within $[1, b^1_{L^1}]$. The search range of each CONV layer is represented as $b^2_i \in \{1, 2, \dots, \lfloor \frac{b^1_i}{cell} \rfloor\}$, where $\lfloor \cdot \rfloor$ denotes the floor function to ensure that the maximum search value for each CONV layer is an integer. That is, 1 in b^2_i represents *cell* channels. The final search structure can be denoted as $B^2 = (b^2_1, b^2_2, \dots, b^2_{L^1})$, whose combination amounts to $\prod_{i=1}^{L^1} \lfloor \frac{b^1_i}{cell} \rfloor$. For example, as shown in Fig. 2, the model structure $B = \{4, 6, 8, 4, 8\}$ is mapped to a search structure $B^2 = \{2, 3, 4, 4\}$ to establish the search space. Using the *cell*-level search space definition approach, the combination for the new search structure B^2 amounts to $\prod_{i=1}^{L^1} \lfloor \frac{b^1_i}{cell} \rfloor$, which reduces the search space formed by the search structure B^1 nearly $(cell)^{L^1}$ times.

In this paper, empirically, the cell size depends on the subset of the minimum number of channels in each layer of the deep neural network. Taking MobileNet-V2 as an example, its minimum channel number is 24, and the corresponding subset is 1, 2, 3, 4, 6, 8, 12, 24. A typical choice for its cell is any value in 2, 3, 4, 6, 8, 12, which both reduces the search space and ensures the search coverage.

4. Experiments

This section conducts extensive experiments across image recognition and object detection tasks to demonstrate the pruning efficiency of FC-ABC. We evaluate image recognition on CIFAR-10, CIFAR-100 and ILSVRC-2012 [29] datasets and object detection on COCO-2017 [30].

For plain networks, we evaluate VGG-16 [24] and GoogLeNet [25]. For coupled networks, we evaluate ResNet-50, ResNet-56/110 [26], DenseNet [31], MobileNet-V2 [27], and ViT-base.

4.1. Implementation details

All experiments are implemented with PyTorch deep learning framework. To improve reproducibility, we follow the data augmentation of PyTorch official example, including random cropping and flipping. We compare Top-1 accuracy, Top-5 accuracy and FLOPs reduction with the existing SOTA methods.

Training Stage: All the networks are trained under SGD with a momentum of 0.9. On CIFAR-10 dataset, for VGG-16 and ResNet-56/110, we set weight decay of 0.0001 and the initial learning rate of 0.1, then it is divided by 10 at 50% and 75% of the total training epochs. For GoogLeNet and MobileNet-V2, we set the weight decay of 0.005 and the initial learning rate of 0.01, then it is divided by 10 at 50% and 75% of the total training epochs. For all these networks on CIFAR-10, we set mini-batch size of 256 and a training epoch of 160. On ImageNet dataset, ResNet-50 is trained for 90 epochs and the learning rate is set 0.01. The batch size is 256.

Searching Stage: During the searching phase, on CIFAR-10, we set the cycles as 5 and the number of candidate structures as 10. On ImageNet and COCO-2017, we set the cycles as 5 and the number of candidate structures as 2. The *cell* settings for each experiment are displayed in the corresponding tables.

Fine-tuning Stage: The momentum is set to 0.9 for all experiments. On CIFAR-10, we iterate 520 epochs with an initial learning rate of 0.1 for VGG-16 and ResNet-56/110. The learning rate is divided by 10 at the 250th and 390th epochs. We iterate 300 epochs with an initial learning rate of 0.01 for GoogLeNet and MobileNet-V2, the learning rate is divided by 10 at the 150th and 225th epochs. On ILSVRC-2012, we train the pruned model 90 epochs with a step scheduler learning rate, which begins at 0.1 and is divided by 10 at 30, 60, and 80 epochs, respectively. On COCO-2017, we train the pruned model 150 epochs.

4.2. Pruning results on ImageNet

ResNet-50: We evaluate its performance on ImageNet and present the results in Table 1. Compared with White-Box and TSGCP, our method achieves the Top-1 accuracy of 75.50% under FLOPs reductions of 50.2%. Compared with DAIS and GAL, our method achieves the Top-1 accuracy of 74.95% under a FLOP reduction of 55.9%. In addition, compared to the ABCPruner, our method results in a more considerable reduction in FLOPs (55.9% versus 54.3%) while maintaining a higher Top-1 accuracy (74.95% versus 73.86%). These results show that the outstanding pruning effectiveness of the proposed method.

MobileNet-V2: Table 1 displays the experimental data, under a 31.7% reduction in FLOPs, our method achieves a Top-1 accuracy of 72.12%. It is noteworthy that compared to the original MobileNet-V2 [27], the Top-1 accuracy has improved by 0.24%. Compared with the several SOTA methods (FSM [38] and UDSP [3]), the pruned model obtained by FC-ABC has the best Top-5 accuracy of 90.29, which is lossless pruning.

ViT-base: Under a 45.5% reduction in FLOPs, our method achieves a Top-1 accuracy of 76.95%. Regarding the selection of 0.4 as the initial pruning ratio, 0.4 represents the median value of the commonly adopted range in prior ViT pruning literature, ensuring comparability.

In sum, on the large-scale dataset, our method demonstrates excellent search effectiveness only on the MobileNet architecture. For ResNet50, it achieves merely satisfactory search performance. This indicates that our approach exhibits certain scalability limitations when dealing with large datasets.

4.3. Pruning results on CIFAR-100

GoogLeNet: As shown in Table 2, for GoogLeNet, compared with the original model [25] and the DepGraph [5], our method achieves the best Top-1 accuracy of 79.13% under FLOP reductions of 15.8%. Additionally, our method results in a more substantial reduction in FLOPs of 65.0% while maintaining a higher Top-1 accuracy of 79.13%.

DenseNet-40: Table 2 displays the experimental data, under a 33.3% reduction in FLOPs, our method achieves a significant pruning ratio for DenseNet-40, resulting in a Top-1 accuracy of 74.37%. Compared with the VCNPN [41], our method results in a more considerable reduction in FLOPs (50.7% versus 37.7%) while maintaining a higher Top-1 accuracy (73.40% versus 72.19%).

In sum, the pruning results on CIFAR-100 dataset can show that FC-ABC has significant pruning efficiency, which demonstrates outstanding search performance on medium-sized datasets. The pruning experiments on models with varying complexities further indicate that our approach maintains strong scalability.

4.4. Pruning results on CIFAR-10

VGG-16: Table 3 shows that, compared with several SOTAs, FC-ABC achieves the best Top-1 accuracy of 93.72% under FLOPs reductions of 80.9%. Moreover, compared to the ABCPruner, which slims networks using an ABC algorithm, FC-ABC results in a more substantial reduction in FLOPs (80.9% versus 73.68%) while maintaining a higher Top-1 accuracy (93.72% versus 93.08%).

GoogLeNet: As shown in Table 3, compared with the SOTA method APIB [45], FC-ABC achieves the Top-1 accuracy of 95.15% and improves the FLOP reductions from 63.0% to 80.07%. Moreover, compared to the ABCPruner, our method results in a more substantial reduction in FLOPs (80.1% versus 66.6%) while maintaining a higher Top-1 accuracy (95.15% versus 94.84%).

ResNet-56/110: Table 3 presents that, for ResNet-56, compared with the original model [26] and the several SOTA methods (DCFF and DepGraph), our method achieves the best Top-1 accuracy of 93.98% under FLOP reductions of 55.21%. Additionally, compared to the ABCPruner, our method results in a more substantial reduction in FLOPs (55.21% versus 54.68%) while maintaining a higher Top-1 accuracy (93.98% versus 93.23%). For ResNet-110, compared with the original model [26] and the several SOTA methods (DCFF [13] and White-Box [32]), our method achieves the best Top-1 accuracy of 94.14% under FLOP reductions of 70.14%. Additionally, compared to the ABCPruner, our method results in a more substantial reduction in FLOPs (70.14% versus 65.04%) while maintaining a higher Top-1 accuracy (94.14% versus 94.12%).

MobileNet-V2: Table 3 displays that, under a 66.3% reduction in FLOPs, our method achieves a significant pruning ratio for MobileNet-V2, resulting in only 0.58M remaining parameters and a Top-1 accuracy of 95.37%. It is noteworthy that compared to the original MobileNet-V2 [27], the Top-1 accuracy has improved by 0.35%. Compared with the several SOTA methods (DDNP [48] and White-Box [32]). Additionally, as can be seen from Table 4, it can be observed that when the pruning ratio is 0.66, the models discovered by our method outperform the baseline significantly. The Top-1 accuracy increases by 0.50% with a 35.6% reduction in FLOPs and a 52.60% reduction of parameter.

In sum, VGG is a plain network in which the convolutional layers are not coupled with dependencies. Therefore, the experimental results on VGG-16 can demonstrate the superiority of the proposed FC-ABC over ABCPruner. Meanwhile, the favorable performance of FC-ABC on models such as ResNet, GoogLeNet, DenseNet, and ViT can verify that the proposed structural mapping strategy effectively addresses the application of FC-ABC to networks with coupled structures.

Table 1
Pruning results of ResNet50 and MobileNet-V2 on ImageNet.

Model	Method	<i>cell</i>	<i>pr</i>	Top-1 Acc. (%)	Top-1 Acc $\uparrow$	Top-5 Acc (%)	Top-5 Acc $\uparrow$	FLOPs $\downarrow$
ResNet-50	LP [9]	–	–	76.15→75.17	–0.98%	92.96→92.35	–0.61%	44.4%
	White-Box [32]	–	–	76.15→75.32	–0.83%	92.96→92.43	–0.53%	45.6%
	TSGCP [33]	–	–	76.15→75.60	–0.55%	92.96→92.58	–0.38%	45.9%
	REAF [34]	–	–	76.15→75.17	–0.98%	92.87→92.44	–0.43%	47.6%
	FC-ABC	1	0.5	76.15→75.50	–0.65%	92.96→92.78	–0.18%	50.2%
	MFP [35]	–	–	76.15→74.86	–1.29%	92.87→92.43	–0.44%	53.5%
	ABCPruner [6]	–	–	76.01→73.86	–2.15%	92.96→91.69	–1.27%	54.3%
	S ² R ² Net[36]	–	–	76.40→75.16	–1.24%	–	–	54.9%
	DAIS [7]	–	–	76.15→74.45	–1.70%	92.87→92.21	–0.66%	55.3%
	BNSC [17]	–	–	76.15→75.30	–0.85%	92.87→92.40	–0.47%	55.7%
	FC-ABC	6.4	0.55	76.15→74.95	–1.20%	92.96→92.38	–0.58%	55.9%
MobileNet-V2	SANP [37]	–	–	71.91→72.05	0.14%	90.30→90.37	0.07%	29.1%
	FSM [38]	–	–	71.88→71.18	–0.70%	90.29→89.81	–0.48%	30.6%
	ATO [39]	–	–	71.88→72.02	0.14%	90.29→90.19	–0.10%	30.1%
	UDSP [3]	–	–	71.88→71.72	–0.16%	–	–	36.6%
	FC-ABC	1	0.3	71.88→72.12	0.24%	90.29→90.29	0.0%	31.7%
ViT-base	CP-ViT [40]	–	–	77.91→77.36	0.55%	–	–	33.5%
	DepGraph [5]	–	–	81.07→79.17	1.90%	–	–	40.9%
	FC-ABC	1	0.4	80.84→76.95	–3.89%	96.02→88.28	–7.74%	45.5%

Table 2
Pruning results of the GoogLeNet and the DenseNet on CIFAR-100.

Method	<i>cell</i>	<i>pr</i>	Top-1 Acc. (%)	Acc. $\uparrow$	FLOPs $\downarrow$
GoogLeNet	–	–	78.51	0.00%	0.0%
FC-ABC	6.4	0.1	78.51→79.91	1.40%	15.8%
DepGraph	–	–	77.56→77.91	0.35%	33.3%
FC-ABC	6.4	0.6	78.51→79.13	0.62%	65.0%
DepGraph	–	–	77.56→77.00	0.56%	66.6%
DenseNet-40	–	–	74.64	0.00%	0.00%
FC-ABC	1	0.3	74.64→74.37	–0.27%	33.3%
VCNNP [41]	–	–	74.64→72.19	–2.45%	37.7%
FC-ABC	1	0.5	74.64→73.40	–1.24%	50.7%

4.5. Pruning results on COCO

We use ResNet50 as the backbone network and adopt SSD300 as the detection head for experiments on the COCO2017 dataset. Table 5 shows that compared to existing methods DMCP and REPrune, the proposed FC-ABC can find the best pruned model, achieving a 50.5% reduction in FLOPs and a 61.5% reduction in parameters, with an mAP of 25.1%. From the experimental results, our model performs well because the discovered network demonstrates optimal performance in detecting small objects, even surpassing the original model. This indicates that our method can uncover superior architectures. Additionally, the search performance in the field of object detection further demonstrates the outstanding generalizability of our method.

4.6. The impact of the computational adaptation under FLOPs constraints

(1) Reducing Fitness Estimation Bias via Local Evaluation Consistency.

FC-ABC leverages local evaluation consistency by grouping similarly structured individuals for evaluation, thereby reducing fitness estimation bias and improving search performance. To verify the local evaluation consistency, we conducted 90 search experiments for structure search using the VGG-16 pre-trained model on CIFAR-10. Among these experiments, we varied the pruning ratio across nine different values ranging from 0.1 to 0.9.

In Fig. 6, the experimental results present the computational cost, fitness value, and final performance of the optimal subnetwork found in each search. The horizontal axis represents the actual computational cost of the pre-trained model, with intervals set at a decay step of 10%. It is evident from the figure that as the computational cost of the sampled subnetwork decreases, the discrepancy between its fitness

Table 3
Pruning results of the plain networks (VGG-16 and GoogLeNet) and the coupled networks (ResNet-56/110 and MobileNet-V2) on CIFAR-10.

Method	<i>cell</i>	<i>pr</i>	Top-1 Acc. (%)	Acc. $\uparrow$	FLOPs $\downarrow$
VGG-16	–	–	93.02	0.00%	0.00%
ABCPruner [6]	–	–	93.02→93.08	0.06%	73.7%
ELC [42]	–	–	93.56→93.11	–0.45%	74.0%
White-Box [32]	–	–	93.02→93.47	0.45%	76.4%
DCFF [13]	–	–	93.02→93.47	0.45%	76.8%
ACPPPO [43]	–	–	93.62→93.47	–0.15%	76.8%
TSGCP [33]	–	–	93.66→93.61	–0.05%	77.3%
FC-ABC	4	0.8	93.47→93.72	0.25%	80.9%
GoogLeNet	–	–	95.05	0.00%	0.00%
CFDP [44]	–	–	95.05→95.25	0.20%	57.2%
APIB [45]	–	–	95.05→95.29	0.24%	63.0%
ABCPruner [6]	–	–	95.05→94.84	–0.21%	66.6%
ACPPPO [43]	–	–	94.74→94.87	0.13%	75.3%
GA [46]	–	–	95.05→95.01	–0.04%	74.4%
FC-ABC	8	0.8	95.05→95.15	0.10%	80.1%
ResNet-56 [26]	–	–	93.26	0.00%	0.0%
REAF [34]	–	–	93.16→93.78	0.62%	38.0%
DepGraph [5]	–	–	93.53→93.64	0.11%	52.6%
ABCPruner [6]	–	–	93.26→93.23	–0.03%	54.1%
ACPPPO [43]	–	–	93.17→93.33	0.16%	61.4%
DCFF [13]	–	–	93.26→93.26	0.00%	55.9%
FC-ABC	6.4	0.5	93.26→93.98	0.72%	55.2%
ResNet-110 [26]	–	–	93.50	0.00%	0.00%
ABCPruner [6]	–	–	93.50→93.58	0.08%	65.0%
Gbp [18]	–	–	93.82→93.71	–0.11%	62.3%
ResPrune [47]	–	–	93.17→93.68	0.51%	63.4%
White-Box [32]	–	–	93.50→94.12	0.62%	66.0%
DCFF [13]	–	–	93.50→93.80	0.30%	66.6%
FC-ABC	–	0.7	93.50→94.14	0.64%	70.1%
MobileNet-V2 [27]	–	–	95.02	0.00%	0.00%
White-Box [32]	–	–	95.02→95.28	0.26%	29.2%
Adapt-DCP [12]	–	–	94.47→94.72	0.25%	34.3%
FC-ABC	1	0.3	95.02→95.52	0.50%	35.6%
DDNP [48]	–	–	94.58→94.81	0.23%	43.0%
FC-ABC	1	0.6	95.02→95.37	0.35%	66.3%

value and its final performance increases markedly. Notably, sampling subnetworks within the same computational cost interval exhibit similar fitness estimation biases. This observation directly supports the rationale behind the local evaluation consistency strategy: by confining the search to a constrained computational cost interval, the sampling

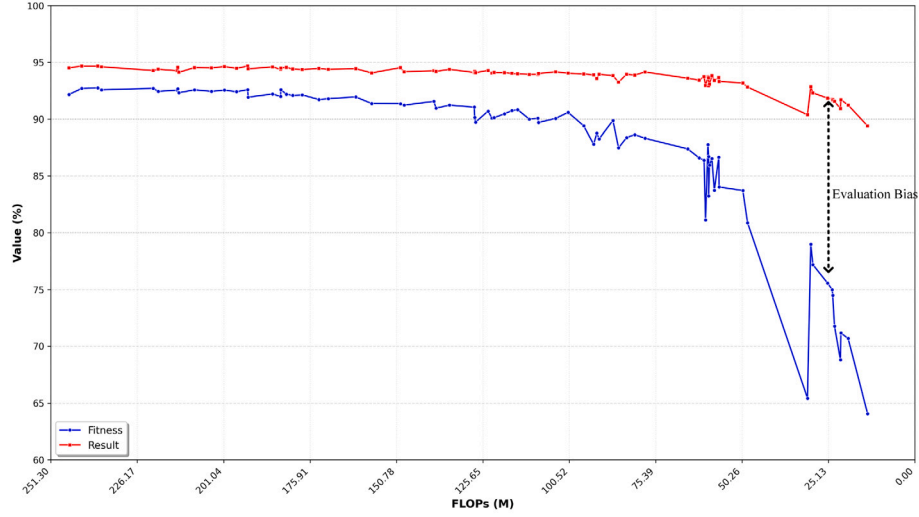


Fig. 6. The parameters and FLOPs of sampled sub-networks during the searching process in ABCPruner and FC-ABC.

Table 4

Pruning results of the MobileNet-V2 on CIFAR-10, with a reduction factor of $cell = 3.2$, which corresponds to a minimum layer search range of $[1, 2, 3, 4, 5]$.

pr	Top-1 Acc. (%)	Acc. $\uparrow$ (%)	FLOPs (M)	Parameters (M)
0.0	95.02	0	321.68	2.30
0.1	95.37	0.35	275.09	1.73
0.2	95.32	0.30	240.85	1.36
0.3	95.52	0.50	206.92	1.09
0.4	95.33	0.31	172.77	1.00
0.5	95.24	0.22	143.71	0.77
0.6	95.37	0.35	108.49	0.58
0.7	94.56	-0.46	96.34	0.44
0.8	94.10	-0.92	61.63	0.40
0.9	91.75	-3.27	30.89	0.17

Table 5

Pruning result of SSD300 with ResNet-50 (the backbone) on COCO-2017. Performance is evaluated using bounding box AP. FLOPs reduction only refers to the ResNet-50 backbone. For FC-ABC, the $cell$ is set as 3, and the pr is set as 0.5.

Method	FLOPs $\downarrow$	mAP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
Baseline	0.0%	25.2	42.7	25.8	7.3	27.1	40.8
DMCP [16]	50.0%	24.1	41.2	24.7	6.7	25.6	39.1
REPrune [2]	50.0%	25.0	42.3	25.9	7.4	26.8	40.4
FC-ABC	50.5%	25.1	42.4	26.0	7.6	26.7	40.3

individuals exhibit similar structural characteristics and thereby share comparable estimation biases.

(2) Empirical Analysis of Sampling Strategies on Fitness-Accuracy Correlation.

To explain why FC-ABC achieves lower fitness-estimation bias than ABCPruner, we analyze the correlation between search-time fitness and post-fine-tuning accuracy under two sampling strategies. The first strategy randomly samples candidates across the entire FLOPs spectrum, mirroring ABCPruner's unconstrained global search. The second strategy samples candidates from contiguous FLOPs regions, mirroring FC-ABC's FLOPs-constrained local search. By comparing these strategies, we aim to demonstrate that the FLOPs constraint reduces bias by confining the search to the region where fitness and accuracy are naturally well-aligned.

The analysis is conducted on the pool of 90 candidate subnetworks of VGG-16/Cifar10. To investigate the relationship between search-space coverage and fitness-estimation bias, we design a comparative

Table 6

PCC comparison between global random sampling (ABCPruner-style) and local continuous sampling across different FLOPs regions and sample sizes.

Sample size	Global random Mean $\pm$ Std	High FLOPs	Mid FLOPs	Low FLOPs
10	0.737 ± 0.112	0.921	0.987	0.811
20	0.853 ± 0.085	0.940	0.991	0.853
30	0.879 ± 0.069	0.940	0.987	0.866
40	0.896 ± 0.060	0.940	0.990	0.876
50	0.908 ± 0.051	0.923	0.980	0.851

experiment that simulates two distinct search behaviors: global random sampling, which randomly selects candidates from the entire FLOPs spectrum and mirrors the unconstrained search strategy of ABCPruner, and local continuous sampling, which selects consecutive candidates from a contiguous FLOPs region and mirrors the FLOPs-constrained search of FC-ABC. To ensure robustness, we evaluate five sample sizes ($k = 10, 20, 30, 40, 50$); for global random sampling, we repeat the random draw 1000 times for each k and report the mean Pearson Correlation Coefficient (PCC) along with the standard deviation to account for sampling variability, while for local continuous sampling, we report the deterministic PCC computed from three distinct FLOPs regions (high, mid, and low) to reveal which region yields the strongest fitness-accuracy alignment. The experimental results are shown in Table 6.

The experimental results reveal three key observations. (i) local continuous sampling from the middle FLOPs region consistently achieves PCC values above 0.98 across all sample sizes $k = 10, 20, 30, 40, 50$, indicating a near-perfect alignment between search-time fitness and post-fine-tuning accuracy with exceptional stability. (ii) global random sampling yields substantially lower PCC values ranging from 0.737 to 0.908, with large standard deviations ($\sigma = 0.051\text{--}0.112$), demonstrating that the fitness-accuracy correlation under global search is not only weaker but also highly unstable and dependent on which candidates are sampled. (iii) not all continuous regions exhibit equally strong correlations: the high FLOPs and low FLOPs regions yield PCC values of approximately 0.92–0.94 and 0.81–0.88, respectively, indicating that even continuous sampling fails if the region falls outside the well-aligned FLOPs interval. These findings lead to two conclusions: (i) the FLOPs constraint in FC-ABC reduces fitness-estimation bias not merely by narrowing the search range, but by confining the search to the specific FLOPs region where fitness and true accuracy are naturally well-aligned; and (ii) ABCPruner's global random sampling forces

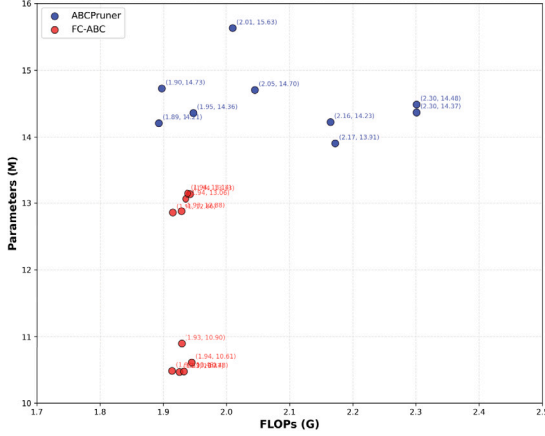


Fig. 7. The parameters and FLOPs of sampled sub-networks during the searching process in ABCPruner and FC-ABC.

the search to traverse incomparable FLOPs levels, introducing systematic bias that results in unreliable fitness evaluations and suboptimal pruning decisions.

(3) The Computation-adaptive Strategy Uses the Local Evaluation Consistency.

To clearly describe the local evaluation consistency for the sub-network's computation adaptive strategy, we adopted ABCPruner and FC-ABC to prune the ResNet-50 model pretrained in ImageNet dataset and the pruning ratio is set as 0.5. And we analyze the distribution of searched sub-networks' FLOPs and Parameters as shown in Fig. 7. It is clear that compared with ABCPruner's global search, every sampled sub-network in FC-ABC meets the real FLOPs' pruning ratio of 0.5 (The FLOPs of ResNet-50 on ImageNet is 4.13 G and the FLOPs threshold for this pruning experiment is 2.065 G). The FLOPs constraint employed in FC-ABC confines the sampled individuals to two concentrated local regions during the search process. This clustering effect enables the search to fully leverage local evaluation consistency, as structurally similar individuals within the same region share comparable fitness estimation biases.

As a result, the negative impact of fitness estimation bias on the final selected subnetwork can be effectively alleviated. Moreover, the sampling network of FC-ABC has a lower actual Parameters count, supported by its computation-adaptive strategy.

4.7. Scalability to diverse network architectures

(1) Handling Coupled Structures via Structural Mapping Strategy.

To describe the structural mapping strategy, we adopted two methods to prune the ResNet-50 model pretrained in ImageNet dataset and the pruning ratio is set as 0.5. As shown in Fig. 8, after pruning with ABCPruner, the channel counts of several convolutional layers remain unchanged from the original model and display a periodic pattern, which can be attributed to the long-distance dependencies among coupled layers. By contrast, the FC-ABC-pruned model also exhibits periodicity, yet with considerably lower peaks. These features that appear periodic in terms of channel numbers are actually coupled structures within the model. Any such structure must ensure equal numbers of channels during the pruning process. It is clear that our method can reduce the channels in couple structures compared with ABCPruner. This improvement is achieved because the structural mapping strategy integrates the coupled structures into the search space, allowing these previously untouchable layers to be effectively pruned.

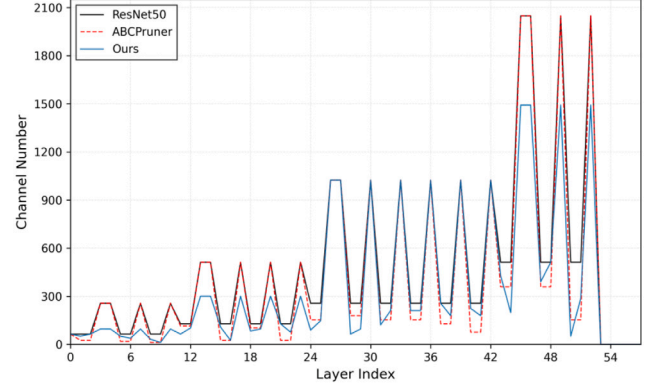


Fig. 8. Comparison of the number of channels of the convolutional layers in ResNet-50 and two pruned models.

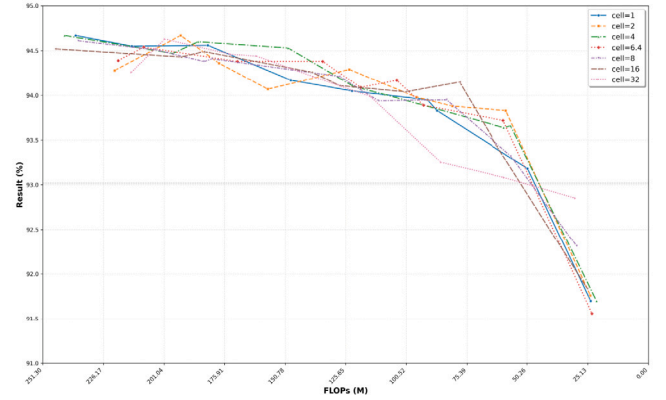


Fig. 9. Searching results of VGG-16 with different cell. Each experimental result is the average of three runs under different FLOPs constraints (0.1, ..., 0.9). The horizontal dashed line represents the baseline.

(2) Search Space Shrinking via Cell-Level Scaling.

To explore the impact of cell-level search space method on the completeness of search space, we conducted search experiments on a VGG-16 pre-trained model on CIFAR-10. We selected 7 different cell values and performed search experiments with 9 different pruning ratios ranging from 0.1 to 0.9. As shown in Fig. 9, across searches with different pruning granularities, when the cell value is 4, relatively better results are obtained at pruning ratios between 0.1 and 0.4. Between 0.4 and 0.8, the seven different cell settings yield comparable results. At a pruning ratio of 0.9, the setting with cell = 32 achieves the best search result. Furthermore, from the overall pruning perspective, all different settings of cell value achieve better results than the pre-trained model across the pruning ratio range of 0.1 to 0.8. The above experiments demonstrate that the proposed FC-ABC can effectively preserve the integrity of the search space. Moreover, compared with the baseline, FC-ABC can remove 80% FLOPs of VGG-16 without a decrease of performance under different cell sizes. The experimental results prove that the cell-level search space definition method does not harm the search space, while significantly reduce the search space and improve search efficiency. Finally, the experiments with different cell values on VGG-16 validate that the proposed cell-level search space reduction method can effectively reduce the search space size without compromising search performance.

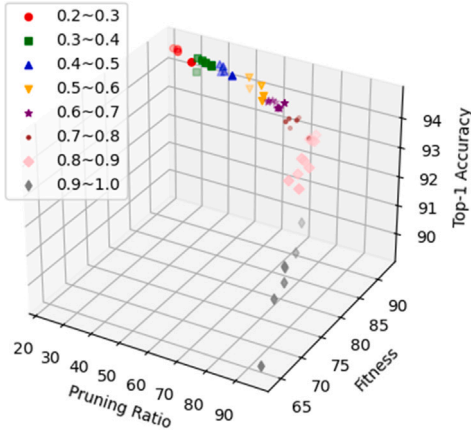


Fig. 10. The model correlation between the fitness and the final accuracy under different pruning ratios of FLOPs.

4.8. The impact of C_1

This section provides an analysis from both theoretical and experimental perspectives. Given search structure B^2 , the FLOPs constraints reduce the search space during search phases. For convenience, assume each channel has the same FLOPs, the reduction ratio of FLOPs can be approximated as the reduction ratio of the combination number of B^2 . Given pruning ratio pr , the FLOPs constraints shrink the size of B^2 to its $\frac{1}{pr}$. Since the main time and computation costs are used to obtain models' fitness during the search process, restricting the entire search space can effectively reduce resource waste, such as the models with wrong FLOPs. Moreover, the FLOPs constraints divides the entire search space into two halves, reducing the ABC's effective search cost by at least pr , significantly enhancing search efficiency.

The correlation experiments are conducted to show the effectiveness of FLOPs in FC-ABC. Some architectures are randomly sampled from VGG-16's search space to get their fitness and final accuracy. The 2-dimensional model correlation is shown in Fig. 1 (Section 1). The ABC performs a global search across the entire search space, whose chaotic correlations makes it difficult to obtain the optimal pruned structure. After inducing the FLOPs constraints, the 3-dimensional model correlation is depicted in Fig. 10. It is obvious that pruned models with similar pruning rates tend to cluster in accuracy, and the pruning ratio of FLOPs separates these different clusters for the searching process. Therefore, FLOPs constraints can effectively reduce the fitness estimation bias and FC-ABC can search the better pruned structure than the original ABC.

4.9. Parameter selection for model fine-tune

In this section, we conduct experiments to illustrate how to select network parameters for pruned models. The baseline model uses ResNet50 as the backbone, combined with an SSD model as the object detection network. The parameters of ResNet50 are pretrained on the ImageNet-1K dataset, while the SSD model parameters are initialized using the Xavier initialization method. For any pruned model, we explored three different parameter selection methods.

As shown in Table 7, if the pruned model is initialized solely using the Xavier initialization method, it yields suboptimal performance. When the parameters pretrained on the ImageNet-1K dataset are combined with the Xavier initialization method for joint initialization, a noticeable performance improvement is observed, but the results are still inferior. Finally, when we use the parameters of the base model, specifically the pretrained model on the COCO-2017 dataset, the pruned model demonstrates optimal performance.

Table 7

The impact of parameter initialization on fine-tuning performance. The pruned model is obtained by FC-ABC, whose $cell$ is set as 3 and the pr is set as 0.5.

Models	ResNet-50	SSD	FLOPs↓	mAP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
Base	ImageNet-1K	Xavier	0.0%	25.2	42.7	25.8	7.3	27.1	40.8
Pruned	Xavier	Xavier	50.5%	12.9	24.4	12.3	2.4	12.1	22.3
Pruned	ImageNet-1K	Xavier	50.5%	20.4	35.5	20.6	5.2	21.0	33.5
Pruned	Base	Base	50.5%	25.1	42.4	26.0	7.6	26.7	40.3

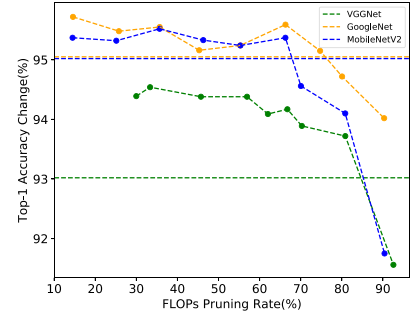


Fig. 11. The comparison of pruning efficiency. Pruning of three models with varying FLOPs constraints (0.1, 0.2, ..., 0.9) was performed on CIFAR-10. The horizontal dashed line represents the baseline of different models.

Table 8

The cost of runtime for different tasks. FOT means that the training time and testing time per epoch on the devices. Food and Time denote the food number and the time consumed during the search process.

Method	Model	Dataset	FOT	M	Food	Time	Devices
FC-ABC	ResNet-56	CIFAR-10	26.8 s+1.92 s	10	10	3.2 h	1080Ti
FC-ABC	GoogLeNet	CIFAR-100	36.5 s+5.5 s	10	10	6.2 h	2*A30
ABCPruner	ResNet-50	ImageNet	0.67 h+45 s	2	2	11.54 h	3090Ti
FC-ABC	ResNet-50	ImageNet	0.67 h+45 s	2	2	11.09 h	3090Ti
FC-ABC	MobileNet-V2	ImageNet	0.24 h+13.2 s	5	10	2.4d	4*A30

4.10. Different computation ranges

To validate the impact of computational constraints C_1 on the search results, we conduct pruning experiments of VGG-16, GoogleNet, and MobileNet-V2 three models on CIFAR-10. Each model is pruned with nine different pruning ratios, and the pruning results are shown in Table 4 and Fig. 11. Fig. 11 demonstrates that the computational constraints are satisfied by the optimal structures, aligning with practical requirements. Compared with the baseline, FC-ABC shows significant pruning performance under different pruning rates. For VGGNet, the optimal structure removes 80.9% of FLOPs with 0.25% improve of Top-1 accuracy. For GoogleNet, the optimal structure removes 80.1% of FLOPs with 0.1% improve of Top-1 accuracy. And for MobileNet-V2, the optimal structure removes 66.3% of FLOPs with 0.35% improve of Top-1 accuracy. Furthermore, it can be observed from both Table 4 and Fig. 11 that when the pruning ratio is low, the lightweight models obtained by our method significantly outperform the baseline, indicating that our approach not only finds lightweight model but also implements structural optimization.

4.11. The analysis of runtime and resource requirements

In this section, we analyze the time and resource costs required for search tasks of varying scales, with specific data shown in Tables 8 and 9. The resources required for a searching task depend on the model and the training dataset. For the searching task with ResNet56 on CIFAR-10, we utilize a 1080Ti GPU, with a total search time of 3.2 h. For the search task with GoogLeNet on CIFAR-100, we utilize two A30 GPUs, resulting in a search time of 6.2 h. For the search task

Table 9

Comparison of pruning time and memory usage for multiple pruning methods. All runtime and memory consumption results in this table were measured directly on the same hardware platform (NVIDIA RTX 3090Ti, CUDA 12.2, PyTorch 2.2) using the official code or our own reimplementation.

Methods	Models	Dataset	Top-1	Pruning time	Pruning memory	Fine-tuning time	Inference time
REAF	ResNet-56	CIFAR10	93.78%	5.53 h	5.55G	0.9 h	1.3 ms
TSGCP	ResNet-56	CIFAR10	93.60%	2.81 h	1.12G	0.9 h	1.2 ms
FC-ABC	ResNet-56	CIFAR10	93.98%	1.61 h	1.02G	0.9 h	1.2 ms
TSGCP	ResNet-50	ImageNet	75.60%	20.5 h	8.12G	18.3 h	18 ms
GA	ResNet-50	ImageNet	73.58%	0.95 h	7.98G	18.3 h	17 ms
ABCPruner	ResNet-50	ImageNet	73.86%	11.54 h	12.6G	18.3 h	20 ms
FC-ABC	ResNet-50	ImageNet	75.50%	11.09 h	12.6G	18.3 h	14 ms

on MobileNet-v2, we utilize four A30 GPUs, with a search time of 2.4 days. Overall, the cost of our search tasks is within an acceptable range. For ResNet-50 pretrained on ImageNet, compared with ABCPruner with a honey value of 16, FC-ABC can search the honey value of 37. After the searching stages, ABCPruner and FC-ABC both search 10 subnetworks, but ABCPruner has searched 1 subnetwork twice. It is clear that FC-ABC can effectively explore a larger number of sub-networks with comparable search time. This benefits from FC-ABC incorporating a subnetwork record set during the search process, which prevents possible redundant fitness calculations with only a negligible time loss, thus enhancing search efficiency.

Moreover, as shown in Table 9, we compare the pruning time, GPU memory consumption, fine-tuning time, and inference time of FC-ABC against GA, TSGCP, REAF, and ABCPruner on both ResNet56/CIFAR-10 and ResNet50/ImageNet. GA is a pruning method based on the evaluation of channel importance. TSGCP is also a method based on the evaluation of channel importance, but it need a auto-sparsity process to enhance the importance of channels. REAF and ABCPruner are both the pruning methods based on structure search. On the more challenging ImageNet benchmark, FC-ABC requires 11.09 h of pruning time—longer than GA (0.95 h) but shorter than TSGCP (20.5 h), and comparable to ABCPruner (11.54 h). This moderate search cost is well justified by the resulting performance: FC-ABC achieves a Top-1 accuracy of 75.50%, substantially outperforming GA (73.58%) and ABCPruner (73.86%), while matching TSGCP (75.60%) with nearly half the pruning time. In terms of GPU memory, FC-ABC consumes 12.6 GB peak during search, primarily due to simultaneous dataset loading and parallel sub-network evaluation; however, excluding the fitness computation, the memory footprint reduces to 4.51 GB, well within the capacity of commonly available GPUs. All compared methods share the same fine-tuning cost (18.3 h on ImageNet), as they require the same number of fine-tuning epochs, but FC-ABC yields the fastest inference time (14 ms) among them, outperforming GA (17 ms), TSGCP (18 ms), and ABCPruner (20 ms).

To provide a more intuitive basis for comparison, we introduce a normalized metric (accuracy gain per unit of search time) relative to the fastest but least accurate baseline (GA). FC-ABC achieves 0.173% per hour, substantially higher than ABCPruner (0.024% per hour) and TSGCP (0.099% per hour), demonstrating that FC-ABC delivers the highest accuracy improvement per unit of search time among structure-search-based methods. In summary, FC-ABC strikes a favorable balance between computational cost and performance benefits: it achieves the best Top-1 accuracy (75.50%) with moderate pruning time, acceptable GPU memory, identical fine-tuning cost to competing methods, and the fastest inference speed. These results confirm that the additional search overhead of FC-ABC is well justified by its superior accuracy, compression, and inference efficiency, reinforcing its practical value for real-world deployment scenarios.

5. Conclusion

In this paper, we designed a scalable computation-adaptive channel pruning method to effectively prune DNNs with different coupled structures. Firstly, structure mapping is proposed to establish

different mapping relationship between coupled model structures in a DNN with non-coupled search structures of the corresponding search space. The proposed structure mapping method can treat the inter-dependent channels in the coupled structure as a whole and prune them simultaneously, thus avoiding the pruning dependency problem caused by the different local importance of these channels and greatly improving the scalability of the pruning method. Secondly, a computation-adaptive artificial bee colony algorithm is designed to search the optimal pruned structure, which not only reduces the search space but also meets practical compression requirements. Finally, a cell-level search space method is proposed to significantly shrink search space. The computation-adaptive artificial bee colony algorithm with cell-level search space method exponentially reduces the search space of the artificial bee colony algorithm, so that for a given DNN network, a good pruning structure can be searched in a short time.

Extensive experiments demonstrate the effectiveness of FC-ABC relative to existing methods, highlighting its practicality for a limited-resource environment. However, for the Transformer-series models, the method presented in this work still possesses substantial potential for theoretical improvement, including aspects such as search space construction, fitness evaluation strategies, and evolutionary search strategies. And the proposed method fails to fully exploit the parallelism of computational tasks during the search process, resulting in a large amount of redundant waiting during the search process. In future work, we will further design algorithms to improve computational parallelism, thereby significantly reducing the time required for a single search task.

CRedit authorship contribution statement

Yijie Chen: Writing – original draft, Methodology, Funding acquisition. **Chengyang Sun:** Methodology. **Yaqi Yao:** Writing – review & editing, Methodology. **Rui Li:** Writing – review & editing, Supervision, Formal analysis. **Renfa Li:** Writing – review & editing, Resources, Methodology. **Keqin Li:** Writing – review & editing, Resources, Methodology.

Declaration of competing interest

The authors declared no potential conflict of interest with respect to the research, authorship, and/or publication of this article. The authors declared that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

Acknowledgments

This work was supported in part by the Natural Science Basic Research Program of Shaanxi under Grants (No. 2025JC-YBQN-888).

Data availability

Data will be made available on request.

References

- [1] W. Deng, K. Li, H. Zhao, A flight arrival time prediction method based on cluster clustering-based modular with deep neural network, *IEEE Trans. Intell. Transp. Syst.* 25 (6) (2023) 6238–6247.
- [2] M. Park, D. Kim, C. Park, Y. Park, G.E. Gong, W.W. Ro, S. Kim, REPrune: Channel pruning via kernel representative selection, *Proc. AAAI Conf. Artif. Intell.* 38 (13) (2024) 14545–14553, <http://dx.doi.org/10.1609/aaai.v38i13.29370>, URL <https://ojs.aaai.org/index.php/AAAI/article/view/29370>.
- [3] S. Gao, Y. Zhang, F. Huang, H. Huang, BilevelPruning: Unified dynamic and static channel pruning for convolutional neural networks, in: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR, 2024*, pp. 16090–16100.
- [4] H. Chen, Y. Li, B. Zheng, T. Chen, Hyperspectral remote sensing image classification based on domain-level complementarity of spatial-spectral component, *Neural Netw.* (2026) 108610.
- [5] G. Fang, X. Ma, M. Song, M.B. Mi, X. Wang, Depgraph: Towards any structural pruning, in: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2023*, pp. 16091–16101.
- [6] M. Lin, R. Ji, Y. Zhang, B. Zhang, Y. Wu, Y. Tian, Channel pruning via automatic structure search, in: *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI-20, 2020*.
- [7] Y. Guan, N. Liu, P. Zhao, Z. Che, K. Bian, Y. Wang, J. Tang, Dais: Automatic channel pruning via differentiable annealing indicator search, *IEEE Trans. Neural Netw. Learn. Syst.* (2022).
- [8] R. Humble, M. Shen, J.A. Latorre, E. Darve, J. Alvarez, Soft masking for cost-constrained channel pruning, in: *European Conference on Computer Vision, Springer, 2022*, pp. 641–657.
- [9] F. Ji, X. Chen, R. Chu, B. Liu, Network slimming using $L_p(p < 1)$ regularization, *Pattern Recognit.* (2025) 111711.
- [10] Z. You, K. Yan, J. Ye, M. Ma, P. Wang, Gate decorator: Global filter pruning method for accelerating deep convolutional neural networks, *Adv. Neural Inf. Process. Syst.* 32 (2019).
- [11] R. Yu, A. Li, C.-F. Chen, J.-H. Lai, V.I. Morariu, X. Han, M. Gao, C.-Y. Lin, L.S. Davis, Nisp: Pruning networks using neuron importance score propagation, in: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2018*, pp. 9194–9203.
- [12] J. Liu, B. Zhuang, Z. Zhuang, Y. Guo, J. Huang, J. Zhu, M. Tan, Discrimination-aware network pruning for deep model compression, *IEEE Trans. Pattern Anal. Mach. Intell.* 44 (8) (2021) 4035–4051.
- [13] M. Lin, B. Chen, F. Chao, R. Ji, Training compact CNNs for image classification using dynamic-coded filter fusion, *IEEE Trans. Pattern Anal. Mach. Intell.* (2023).
- [14] T. Elsken, J.H. Metzen, F. Hutter, Neural architecture search: A survey, *J. Mach. Learn. Res.* 20 (55) (2019) 1–21.
- [15] Z. Hou, M. Qin, F. Sun, X. Ma, K. Yuan, Y. Xu, Y.-K. Chen, R. Jin, Y. Xie, S.-Y. Kung, CHEX: Channel exploration for CNN model compression, in: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR, 2022*, pp. 12287–12298.
- [16] S. Guo, Y. Wang, Q. Li, J. Yan, DMCP: Differentiable Markov channel pruning for neural networks, in: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR, 2020*.
- [17] F. Hu, J. Zhang, S. Gao, Y. Lin, W. Zhou, R. Wang, An efficient training-from-scratch framework with BN-based structural compressor, *Pattern Recognit.* 153 (2024) 110546.
- [18] X. Lu, W. Dong, Z. Fang, J. Lin, X. Li, G. Shi, Growing-before-pruning: A progressive neural architecture search strategy via group sparsity and deterministic annealing, *Pattern Recognit.* (2025) 111697.
- [19] H. Zhao, L. Li, W. Deng, Multi-UAV path planning using improved artificial hummingbird algorithm based on differential evolution and gradient descent, *IEEE Trans. Consum. Electron.* (2025).
- [20] Z. Tang, Z. Xie, X. Wen, B. Tang, W. Song, EvoDiff-NAS: Evolutionary neural architecture search with diffusion-guided mutation, *Swarm Evol. Comput.* 107 (2026) 102442.
- [21] J. Tong, W. Jiang, J. Zou, Y. Xia, G-evonas: Evolutionary neural architecture search based on network growth, *Swarm Evol. Comput.* 108 (2026) 102483.
- [22] M. Ma, Y. Wang, H. Li, L. Meng, B. Zhang, A reinforcement learning-integrated discrete artificial bee colony algorithm for hybrid flow shop scheduling with batch processing and variable subplot sizes, *Swarm Evol. Comput.* 100 (2026) 102253.
- [23] Y. Lu, C. Wang, X. Gu, X. Pan, R. Zhang, A multi-stage artificial bee colony clustering algorithm, *Swarm Evol. Comput.* 107 (2026) 102436.
- [24] K. Simonyan, A. Zisserman, Very deep convolutional networks for large-scale image recognition, 2014, arXiv preprint arXiv:1409.1556.
- [25] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, A. Rabinovich, Going deeper with convolutions, in: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2015*, pp. 1–9.
- [26] K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, in: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016*, pp. 770–778.
- [27] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, L.-C. Chen, Mobilenetv2: Inverted residuals and linear bottlenecks, in: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2018*, pp. 4510–4520.
- [28] H. Li, A. Kadav, I. Durdanovic, H. Samet, H.P. Graf, Pruning filters for efficient convnets, 2016, arXiv preprint arXiv:1608.08710.
- [29] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, et al., Imagenet large scale visual recognition challenge, *Int. J. Comput. Vis.* 115 (2015) 211–252.
- [30] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, C.L. Zitnick, Microsoft coco: Common objects in context, in: *Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6–12, 2014, Proceedings, Part V 13*, Springer, 2014, pp. 740–755.
- [31] G. Huang, Z. Liu, L. Van Der Maaten, K.Q. Weinberger, Densely connected convolutional networks, in: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2017*, pp. 4700–4708.
- [32] Y. Zhang, M. Lin, C.-W. Lin, J. Chen, Y. Wu, Y. Tian, R. Ji, Carrying out cnn channel pruning in a white box, *IEEE Trans. Neural Netw. Learn. Syst.* (2022).
- [33] Y. Chen, R. Li, W. Li, J. Wang, R. Li, Three-stage global channel pruning for resources-limited platform, *IEEE Trans. Neural Netw. Learn. Syst.* (2023).
- [34] X. Zhang, W. Xie, Y. Li, K. Jiang, L. Fang, Reaf: Remembering enhancement and entropy-based asymptotic forgetting for filter pruning, *IEEE Trans. Image Process.* (2023).
- [35] Y. He, P. Liu, L. Zhu, Y. Yang, Filter pruning by switching to neighboring CNNs with good attributes, *IEEE Trans. Neural Netw. Learn. Syst.* (2022).
- [36] Y. Yan, X. Li, S²R²Net: A network architecture for efficient inference with sample-specific redundancy reduction, *Pattern Recognit.* (2025) 111831.
- [37] S. Gao, Z. Zhang, Y. Zhang, F. Huang, H. Huang, Structural alignment for network pruning through partial regularization, in: *Proceedings of the IEEE/CVF International Conference on Computer Vision, ICCV, 2023*, pp. 17402–17412.
- [38] Y. Duan, Y. Zhou, P. He, Q. Liu, S. Duan, X. Hu, Network pruning via feature shift minimization, in: *Proceedings of the Asian Conference on Computer Vision, ACCV, 2022*, pp. 4044–4060.
- [39] X. Wu, S. Gao, Z. Zhang, Z. Li, R. Bao, Y. Zhang, X. Wang, H. Huang, Auto-train-once: Controller network guided automatic network pruning from scratch, in: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2024*, pp. 16163–16173.
- [40] Z. Song, Y. Xu, Z. He, L. Jiang, N. Jing, X. Liang, Cp-vit: Cascade vision transformer pruning via progressive sparsity prediction, 2022, arXiv preprint arXiv:2203.04570.
- [41] C. Zhao, B. Ni, J. Zhang, Q. Zhao, W. Zhang, Q. Tian, Variational convolutional neural network pruning, in: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2019*, pp. 2780–2789.
- [42] J. Wu, D. Zhu, L. Fang, Y. Deng, Z. Zhong, Efficient layer compression without pruning, *IEEE Trans. Image Process.* 32 (2023) 4689–4700, <http://dx.doi.org/10.1109/TIP.2023.3302519>.
- [43] Y. Hu, D. Chen, F. Zou, Y. Liu, Automatic channel pruning by neural network based on improved poplar optimisation algorithm, *Knowl.-Based Syst.* 310 (2025) 113002, <http://dx.doi.org/10.1016/j.knsys.2025.113002>.
- [44] S. Khaki, W. Luo, CFDP: Common frequency domain pruning, in: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops, 2023*, pp. 4714–4723.
- [45] S. Guo, L. Zhang, X. Zheng, Y. Wang, Y. Li, F. Chao, C. Wu, S. Zhang, R. Ji, Automatic network pruning via Hilbert-Schmidt independence criterion lasso under information bottleneck principle, in: *Proceedings of the IEEE/CVF International Conference on Computer Vision, ICCV, 2023*, pp. 17458–17469.
- [46] M. Saffar, A. Kalhor, A. Habibnia, A geometric approach for accelerating neural networks designed for classification problems, *Sci. Rep.* 14 (1) (2024) 17590.
- [47] A. Jayasimhan, P. Pabitha, ResPrune: An energy-efficient restorative filter pruning method using stochastic optimization for accelerating CNN, *Pattern Recognit.* 155 (2024) 110671.
- [48] S. Gao, F. Huang, Y. Zhang, H. Huang, Disentangled differentiable network pruning, in: *European Conference on Computer Vision, Springer, 2022*, pp. 328–345.