Improved Differentiable Architecture Search With Multi-Stage Progressive Partial Channel Connections

Yu Xue, *Member, IEEE*, Changchang Lu, Ferrante Neri, *Senior Member, IEEE*, and Jiafeng Qin

***Abstract*—Neural architecture search has attracted great attention in the research community and has been successfully applied in the industry recently. Differentiable architecture search (DARTS) is an efficient architecture search method. However, the networks searched by DARTS are often unstable due to the large gap in the architecture depth between the search phase and the verification phase. In addition, due to unfair exclusive competition between different candidate operations, DARTS is prone to skip connection aggregation, which may cause performance collapse. In this article, we propose progressive partial channel connections based on channel attention for differentiable architecture search (PA-DARTS) to solve the above problems. In the early stage of searching, we only select a few key channels for convolution using channel attention and reserve all candidate operations. As the search progresses, we gradually increase the number of channels and eliminate unpromising candidate operations to ensure that the search phase and verification phase are all carried out on 20 cells. Due to the existence of the partial channel connections based on channel attention, we can eliminate the unfair competition between operations and increase the stability of PA-DARTS. Experimental results showed that PA-DARTS could achieve 97.59% and 83.61% classification accuracy on CIFAR-10 and CIFAR-100, respectively. On ImageNet, our algorithm achieved 75.3% classification accuracy.**

***Index Terms*—Differentiable architecture search, exclusive competition, channel attention, progressive partial channel connections.**

I. Introduction

In RECENT years, deep neural networks have been widely used in the industry due to their outstanding performance [1], [2]. The topological structure of deep neural networks plays a decisive role in their performance. Early neural network architectures were mostly designed manually by professionals, such as VGG [3], Resnet [4] and GoogleNet [5]. Although these networks have performed impressively, designing an excellent deep neural network architecture requires plenty of trial and error by experts. In modern industry, a large number of neural networks that adapt to different tasks are required to be designed every day, and completely relying on manual design to accomplish this work is impractical. Because of this, neural architecture search (NAS), an important part of automated machine learning, has been proposed and has received extensive attention.

In recent years, NAS has made great progress. Reinforcement learning (RL) [6] and evolutionary computing (EC) [7] are two widely used methods for NAS. The RL-based method regards NAS as the process of an agent's action. The action space is the same as the neural network search space. The evaluation performance of the neural network is regarded as the reward of the agent. Zoph et al. [8] were the first to use RL to search for the optimal neural network architecture. They used a recurrent neural network as an agent and encoded the neural network architecture into a variable-length string and then used the agent to generate the string. Because each generated architecture needs to be trained from scratch, RL-based methods often require abundant computing resources. For example, Zoph et al. [8] achieved a classification error rate of 3.65% on CIFAR-10, but the entire search process required 800 graphics processing units (GPUs) to search continuously for three weeks. Huge resource consumption means that it is difficult for this method to be widely used. EC [9], [10] is another method that is often used to search for the optimal neural network architecture. This method encodes each network architecture into an individual of a generation. In the encoding process, fixed-length coding or variable-length coding can be used. Although variable-length coding can increase the diversity of the searched architecture, it increases spatial complexity. In the search process, new network architectures are explored through selection, crossover and mutation. Using EC to search for optimal neural network architectures can effectively prevent the search from getting stuck in a local optimum. Sun et al. [11] used genetic algorithms to search for new neural network architectures with the basic blocks from ResNet and DenseNet and achieved good results. Xue et al. [12] used an adjacency matrix to represent candidate architectures and used multi-objective optimisation techniques to search for the optimal architecture. However, using RL-based methods or evolutionary techniques to search for neural network architectures requires independent training to evaluate each newly generated individual, which requires considerable time. For example, Real et al. [13] used genetic algorithms to search for the optimal neural network architecture on CIFAR-10. They finally achieved a classification accuracy of 96.88%, but the entire search process took seven days with 450 GPUs.

Manuscript received 28 October 2022; revised 7 May 2023 and 19 May 2023; accepted 22 June 2023. Date of publication 17 August 2023; date of current version 23 January 2024. This work was supported in part by the National Natural Science Foundation of China under Grants 61876089, 61876185, and 61902281, in part by the Natural Science Foundation of Jiangsu Province under Grant BK20141005, and in part by the project Distinguished Professors of Jiangsu Province. *(Corresponding author: Yu Xue.)*

Yu Xue, Changchang Lu, and Jiafeng Qin are with the School of Software, Nanjing University of Information Science and Technology, Nanjing 211544, China (e-mail: xueyu@nuist.edu.cn; 20211221025@nuist.edu.cn; qinjiafeng@nuist.edu.cn).

Ferrante Neri is with the School of Software, Nanjing University of Information Science and Technology, Nanjing 211544, China, and also with the Department of Computer Science, University of Surrey, GU2 7XH Guildford, U.K. (e-mail: f.neri@surrey.ac.uk).

Digital Object Identifier 10.1109/TETCI.2023.3301395

The most serious problem with using RL or EC in searching for neural architectures is that each searched architecture needs to be independently trained and evaluated, which consumes an immense amount of resources. To solve this problem, researchers have proposed many methods, such as weight inheritance [14], low fidelity estimation [15], curve extrapolation [16] and weight sharing [17]. However, achieving satisfactory results with these methods is often difficult due to their subjectivity. Recently, surrogate models have been used to reduce the amount of computation. This method uses a trained predictor to evaluate the performance of a neural network architecture and has shown good performance. For example, Sun et al. [18] took a random forest as a proxy model to evaluate the performance of a neural network architecture, thus avoiding extensive training. Tang et al. [19] predicted the performance of neural network architectures directly with a semi-supervised predictor. One-shot NAS is another neural network architecture search method, which regards each possible network architecture in the search space as a subnet of a supernet. Each subnet directly inherits the weight of the supernet. This way, only the supernet needs to be trained in the search process, which greatly reduces the search time. For example, SMASH [20] uses a supernet to guide the sorting of subnets, speeding up the evaluation process of the subnets.

To further improve search efficiency, differentiable architectural search (DARTS) [21] continuously relaxes the search space so that gradient methods can be used to alternately optimise the architecture parameters and weight parameters of the supernet. Although DARTS can accelerate the search process for neural network architectures, it also has some defects. First, DARTS has a large gap in the architecture depth between the search phase and the verification phase. In the search phase, the parameters of all operations need to be inputted into GPU memory, which leads to DARTS searching on eight layers but evaluating on 20 layers. While a cell that performs well on a shallow network may not achieve good results on a deep network. Second, due to unfair competition between different operations, weight-free operators, such as skip connections, tend to dominate the architecture, leading to poor performance in the final neural network architecture.

To alleviate these issues, in this article, we propose a progressive partial channel connections strategy based on channel attention for differentiable architecture search (PA-DARTS). In the early stage of the search, we use a channel attention mechanism to select a small number of key channels for the convolution operation, which not only reduces GPU memory consumption but also alleviates unfair competition between the different operations. As the search proceeds, we gradually increase the number of chosen channels to reduce information loss and increase search stability. By increasing the selected channels, we gradually minimise the unpromising candidate operations. Through this strategy, we can ensure that the entire search process is always carried out on 20 cells. We summarise the contributions of this article as follows:

1) We have applied an attention mechanism for channel selection. During the search process, some key channels are selected for partial channel connections using channel attention. The selected channels are sent into the operation space, while the others are directly concatenated with the output of the operation space. This method reduces the unfair competitive advantage of weight-free operations over weight-equipped operations caused by insufficient training.
2) We have proposed a progressive partial channel connections strategy. As the search proceeds, we gradually increase the number of selected channels. At the last stage of the search, we select all channels to avoid the decline in final accuracy caused by information loss.
3) In the search process, we have used a progressive operation pruning strategy to prune unpromising operations. By combining operation pruning with the progressive partial channel connections strategy, we can ensure that the entire search process is always carried out on 20 cells, thus eliminating the gap in the architecture depth between the search phase and the verification phase.
4) We have verified the performance of our proposed algorithm on the CIFAR-10, CIFAR-100 and ImageNet datasets. Experiments have shown that the optimal network architecture searched by PA-DARTS achieved 97.59% classification accuracy on CIFAR10 and 83.62% classification accuracy on CIFAR-100. We have also transferred the architecture searched on CIFAR-10 to ImageNet and achieved 75.3% classification accuracy.

The rest of the article is organised as follows. In Section II, we provide a brief review of DARTS and introduce its main defects. Section III introduces the progressive partial channel connections based on channel attention in detail and the relationship of our work to prior works. In Section IV, we explain our experimental datasets, experimental details, experimental results and ablation experiments. Finally, we provide our conclusion in Section V.

II. Related Works

A. Differentiable Architecture Search

DARTS uses a gradient-guided search method to search for the optimal neural architecture, which greatly reduces complexity as well as achieves good results. The search space of DARTS is composed of several cells. Each cell is composed of two input nodes, one output node and several intermediate nodes. The input of each intermediate node is calculated based on all of its predecessor nodes, as shown in formula (1):

$$x^{(j)} = \sum_{i<j} o^{(i,j)}(x^{(i)}) \tag{1}$$

where $x^{(i)}$ represents the output of intermediate node i, and $o^{(i,j)}$ represents the mixed operation between node i and node j.

The search process of DARTS is divided into two phases: weight optimisation and architecture optimisation. The weight parameter ω is optimised by the training loss L_{train}, and the valid loss L_{val} is used to optimise the architecture parameter $\alpha = \{\alpha^{(i,j)}\}$. The optimisation process of DARTS can be expressed by formulas (2–3):

$$\min_{\alpha} \quad L_{val}(\omega^*(\alpha), \alpha) \tag{2}$$

$$\text{s.t.} \quad \omega^*(\alpha) = \arg\min_{\omega} L_{train}(\omega, \alpha) \tag{3}$$

In DARTS, the weight parameters ω and the architecture parameters α are optimised alternately. To optimise the architecture parameters using the gradient descent method, as shown in formula (4), DARTS uses softmax to make the search space continuous:

$$o^{(i,j)}(x) = \sum_{o \in O} \frac{\exp\left(\alpha_o^{(i,j)}\right)}{\sum_{o' \in O} \exp\left(\alpha_{o'}^{(i,j)}\right)} \tag{4}$$

where O represents the candidate operation space, and $\alpha_o^{(i,j)}$ represents the architecture parameter value of candidate operation o between node i and node j.

After the search, the top-k operations with the highest weights (from distinct nodes) are selected from all candidate operations by formula (5), and the others are removed:

$$o_{select}^{(i,j)} = \arg\max_{o \in O} \alpha_o^{(i,j)} \tag{5}$$

B. Bottlenecks of Differentiable Architecture Search

Numerous experimental results have shown that DARTS suffers from the aggregation of skip connections, which leads to poor performance. Xu et al. [22] analysed this phenomenon and pointed out that the reason is unfair competition between weight-free operations (e.g. skip-connect and max pooling) and weight-equipped operations (e.g. convolution operations). In the early stage of searching, weight-free operations produce more consistent information, while weight-equipped operations propagate inconsistent information due to a large number of untrained parameters. As a result, weight-free operations accumulate many unfair advantages in the early stage of the search, which makes it difficult for weight-equipped operations to defeat them. To solve this problem, Real et al. proposed a partial channel connections strategy in PC-DARTS, which selects fewer channels randomly and sends them into the mixed computation of $|O|$ operations, while other channels directly skip these operations. This strategy reduces the difference between weight-free operations and weight-equipped operations. Other scholars have also provided many solutions to solve this problem. For example, P-DARTS [23] and DARTS+ [24] directly select a fixed number of skip connections with the largest weights. Although this manual selection operation is very intuitive, it is challenging to explain the reason. FairDARTS [25] uses the sigmoid function instead of softmax to eliminate the unfair competition between operations. In PRDARTS [26], a binary gate is introduced for each operation to relieve unfair competition. In this article, we propose a progressive partial channel connections strategy based on channel attention. In the early stage of searching, we only select some key channels to send into the mixed computation of $|O|$ operations using a channel attention mechanism. As the weight-equipped operations continue to be trained, we gradually increase the number of channels selected. This method not only solves the problem of skip connection aggregation but also alleviates the information loss caused by partial channel sampling.

During the search process, DARTS needs to input the parameters of all the candidate operations into the GPU, which leads to an explosion of GPU memory. To alleviate this problem, DARTS stacks eight cells in the search phase and 20 cells in the verification phase, which leads to a large gap in the architecture depth between the search phase and the verification phase, thus affecting the performance of the final network architecture. P-DARTS [23] divides the entire search process into three stages and increases the number of layers progressively using an operating pruning strategy. However, the number of network layers in each stage is inconsistent, and the depth gap still exists in the search process. StacNAS [27] selects a fitting depth in the training phase using gradient confusion. To save GPU memory, GDAS [28] only chooses part of the operations for optimisation, and these operations are input into GPU memory during training. In this article, we combine the progressive operation pruning strategy with the progressive partial channel connections strategy. We gradually prune the unpromising operations as the selected channels are increased to ensure that the search process is always carried out on 20 cells. Our algorithm solves the problem of the depth gap.

C. Attention Mechanism in Neural Architecture Search

An attention mechanism enables neural networks to achieve better performance by selectively processing information of interest. A large number of computer vision tasks incorporate attention modules due to the excellent performance of attention mechanisms. SENet [29] is a classic work with a channel attention mechanism, which obtains the interdependence between channels through a squeeze-and-extraction block and then enhances useful features and suppresses useless features. CBAM [30] uses an attention mechanism module that combines spatial attention and channel attention and achieves better results compared with SENet. Due to the excellent performance of attention mechanisms, many scholars have attempted to introduce them into NAS. The search-based spatial and channel joint attention module (NAS-SCAM) [31] was the first work that directed attention to NAS. This new attention module was applied to the process of architecture search. Based on SENet, AGNAS [32] added an attention module to each candidate operation of the supernet to improve the representation ability of the supernet. Xue et al. introduced a channel attention mechanism to partial channel selection for the first time with ADARTS [33] and achieved good classification results. G-DARTS-A [34] divides the channels into several groups based on PC-DARTS and assigns a separate weight to each group through attention mechanisms. The successful application of these attention mechanisms has promoted the development of NAS.

III. PA-DARTS: Progressive Differentiable Architecture Search Based on Channel Attention

Due to the unfair competition between operations, DARTS is prone to skip connection aggregation. In addition, because the information in each node must be saved in GPU memory during the search process, DARTS requires a large amount of GPU memory; therefore, it cannot directly search on 20 cells. In this section, we introduce a progressive partial channel connections strategy based on channel attention in detail. Not only can the proposed method solve the problem of weight-free operation aggregation, but it also enables the whole search process to run

Algorithm 1: Framework of PA-DARTS.

Require: Training stage N, epochs in each stage P, selected channel proportion $1/k_t$ $(1 \leq t \leq n)$, number of candidate operations reserved in each stage O_t $(1 \leq t \leq n)$.
Ensure: the best network architecture α

```
1:  1 ⇐ i
2:  while i ≤ N do
3:    Build the network according to the proportion of
      channels selected 1/k_i.
4:    if i > 1 then
5:      The network inherits the weight of stage i − 1
6:    end if
7:    1 ⇐ j
8:    while j ≤ P do
9:      Apply mixed calculations on candidate operations
        using Formula (8) with 1/k_i channel selected.
10:     Update architecture α by descending
        ∇L_val(ω − ξ∇_ω L_train(ω, α), α)
11:     Update weights ω by descending ∇_ω L_train(ω, α).
12:     j ⇐ j + 1
13:   end while
14:   reserve the O_{i+1} most promising operations and
      prune other operations
15:   i ⇐ i + 1
16: end while
17: Derive the best architecture α according to Formula
    (5).
18: return the best architecture α
```

on 20 cells to solve the depth gap between the search phase and the verification phase. In the following subsections, we explain the overall framework of PA-DARTS, the progressive partial channel connections strategy based on channel attention and the operation pruning strategy.

A. Overall Framework

Our algorithm is divided into n stages. In the first stage, we retain all candidate operations and select $1/k_1$ proportional channels based on the attention mechanism for partial channel connections. In the intermediate phase t, we retain l_t $(l_t < l_{t-1})$ most promising candidate operations and select $1/k_t$ $(k_t > k_{t-1})$ proportional channels for partial channel connections based on the attention mechanism. In the last stage n, we reserve all channels. Fig. 1 illustrates the overall framework of our algorithm with three stages. We select 1/4 proportional channels for partial channel connections and retain all candidate operations in stage 1. In stage 2, we retain the three most promising candidate operations and select 1/2 proportional channels for partial channel connections. In the last stage, we retain the two most promising candidate operations and reserve all channels. The pseudo codes of PA-DARTS are provided in Algorithm 1. As the parameters of the weight-equipped operations are continuously trained, we gradually increase the number of selected channels. This strategy not only solves the problem of unfair competition among the different operations but also alleviates the loss of channel information and enhances the stability of the algorithm. To carry out the entire search process on 20 cells, we introduce an operation pruning strategy as the number of channels is increased.

B. Partial Channel Connections Based on Channel Attention

The partial channel connections strategy proposed in PC-DARTS can alleviate the aggregation of skip connections well, but channel sampling in PC-DARTS is random. This not only leads to the loss of some key information but also makes the algorithm unstable; thus, it requires additional edge normalisation operations. In this article, the proposed algorithm automatically acquires the importance of each channel through learning, then we enhance the useful features and suppress the useless features for the current task according to the learned importance. This idea is similar to the channel attention mechanism proposed in SENet [29]. Like CBAM [30], assuming that the input feature F has C channels with width W and height H, we can calculate the channel attention map M by formula (6):

$$\begin{aligned} M_c(F) &= \sigma(MLP(AvgPool(F)) + MLP(MaxPool(F))) \\ &= \sigma(\omega_1(\omega_0(F^v_{avg})) + \omega_1(\omega_0(F^v_{\max}))) \end{aligned} \tag{6}$$

where σ represents the sigmoid activation function, ω_0 is the parameter between the input layer and the hidden layer, and ω_1 is the parameter between the hidden layer and the output layer of the multi-layer perceptron. Through the above operations, we get a tensor $M_c(F)$ containing the channel attention with a dimension of $1 \times 1 \times C$, and the channel with a higher attention value often transmits more important information. By multiplying $M_c(F)$ by F according to the channel order, we can obtain new feature maps $M'_c(F)$:

$$M'_c(F) = M_c(F) * F \tag{7}$$

During the convolution calculation process, the importance of the information on each channel varies. In PC-DARTS, prioritising the channels with key information as much as possible when selecting channels not only improves the accuracy but also increases the stability of the search. Based on this, we introduce a channel attention mechanism to the process of partial channel connections. We select $1/k$ channels with the highest attention values, which are marked as 1, while the other channels are marked as 0. Then, the channels marked as 1 are placed into the operation space, while the other channels bypass the operation space and are directly copied to the output of the operation space. The pseudo codes of the partial channel connections based on the attention mechanism are provided in Algorithm 2.

By introducing the attention mechanism to the partial channel connections, we only select the $1/k$ channels with the largest amount of information and send them into the operation space for computing, which greatly reduces the GPU memory required in the search process. Thus, our method can directly search on 20 cells. At the same time, we use the attention mechanism in the process of selecting channels, which ensures that the channels containing key information are selected to the greatest extent to increase the stability of the proposed algorithm. The

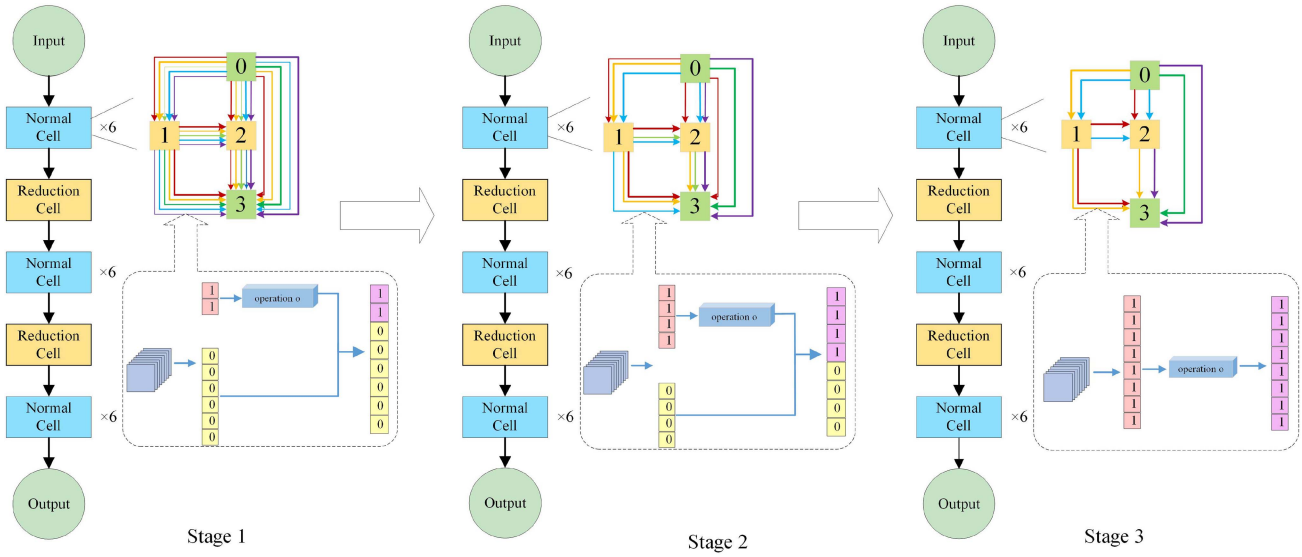


Fig. 1. Framework of the progressive partial channel connections. The entire algorithm is divided into three stages, and each stage directly searches on 20 cells. Partial channel connections based on channel attention is used except for the last stage. The number of channels selected is gradually increased, and the unpromising candidate operations are pruned as the search proceeds.

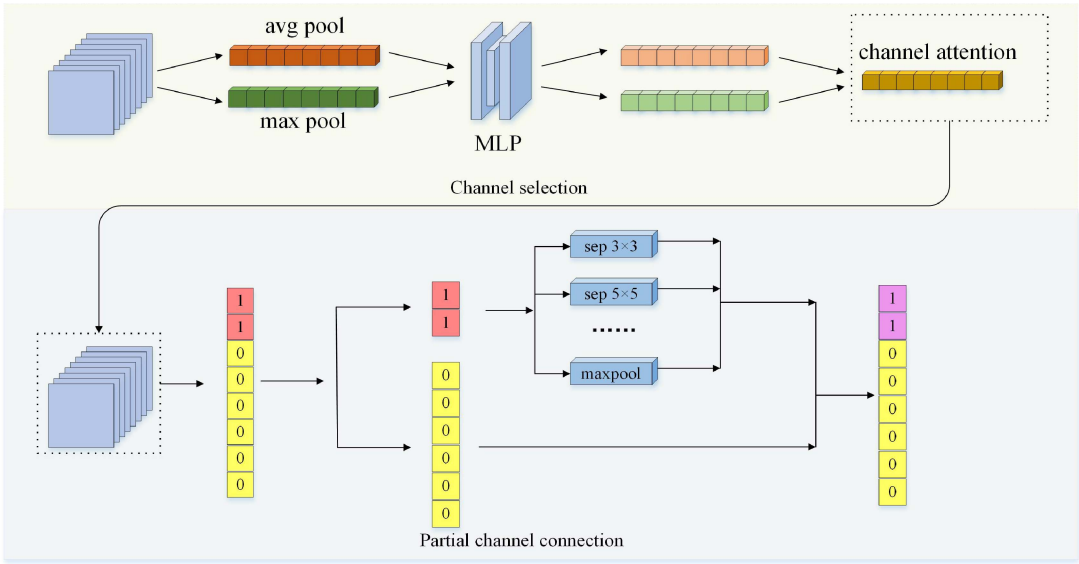


Fig. 2. Process of the partial channel connections. First, the attention map is obtained through the channel attention mechanism, then the channel with high attention is selected and sent to the operation space, while the other channels skip the operation space directly.

entire process of partial channel connections based on channel attention is shown in Fig. 2.

C. Progressive Partial Channel Connections

The partial channel connections strategy not only reduces the computation load but also resolves the unfair competition between weight-free operations and weight-equipped operations. Due to the insufficient training of the weight-equipped operations in the early stage of the search, they tend to propagate inconsistent information, which leads to the accumulation of a large number of disadvantages. In partial channel connections strategies, only a subset of all channels is chosen for computing, thus reducing the difference between weight-free and weight-equipped operations and weakening the advantages of weight-free operations.

As the search proceeds, the parameters of the weight-equipped operations are continuously optimised, and the advantages of the weight-free operations are gradually reduced until the unfair competition between the weight-free and weight-equipped operations is eliminated. Although the channel selection strategy based on channel attention allows for the selection of channels with key information to the greatest extent possible, selecting a small number of channels for partial channel connections in the entire search process is likely to result in the loss of plenty of useful feature information. Considering that the competitive

Algorithm 2: Partial Channel Connections Based on Attention Mechanism.

Require: Feature data F with c channels, selected channel proportion $1/k$.
Ensure: mixed computation $o(x_j)$ of node i

$1 \Leftarrow i$
while $i \leq N$ **do**
  $M_c(F) \Leftarrow \sigma(MLP(AvgPool(F)) + MLP(MaxPool(F)))$
  $M'_c(F) \Leftarrow M_c(F) * F$
  $M_{\max} \Leftarrow$ top $1/k$ weight in $M_c(F)$
  $i \Leftarrow 0$
  while $i < c$ **do**
    if $M_c(F_i) \in M_{\max}$ **then**
      $M'_c(F_i)$ is sent to operation space for mixed computing.
    else
      $M'_c(F_i)$ skips the operation space.
    end if
    $i \Leftarrow i + 1$
  end while
end while
The feature output through the operation space and the unprocessed feature are contacted to produce $o(x_j)$ according to the original channel order.
return $o(x_j)$

advantage of weight-free operations against weight-equipped operations decreases in the later search period, we propose a progressive partial channel connections strategy. We divide the search process into n stages and gradually increase the number of channels selected during the search process. We choose $1/k_t$ of all channels in stage t and choose $1/k_{t+1}$ of the channels in stage $t+1$ for the convolution operation. The number of channels selected in stage $t+1$ is greater than that selected in stage t. The mixed computation of stage t is defined as formula (8):

$$f_{AP}^{(i,j)}(x_i, att_{ij}^t) = \sum_{o \in O} \frac{\exp\left(\alpha_o^{(i,j)}\right)}{\sum_{o' \in O} \exp\left(\alpha_{o'}^{(i,j)}\right)} \cdot o\left(att_{ij}^t(x_i)\right) + \left(1 - att_{ij}^t(x_i)\right) \tag{8}$$

where $att_{ij}^t(x_i)$ refers to the selected channels using channel attention in stage t. Stage $t+1$ inherits the weight parameter of stage t, which can ensure that the parameters of the candidate operations of stage $t+1$ have been trained for t stages. After a certain amount of training, the unfair competition between the weight-free and weight-equipped operations is weakened. At this time, we can select more channels and send them into the operation space. In the last stage, we select all the channels. This operation not only solves the problem of weight-free operation aggregation in the search process but also the problem of information loss, thus increasing the stability of the algorithm.

D. Progressive Operation Pruning

Due to the gradual increase in the number of channels, a sharp increase in GPU memory in the training process occurs. When all channels are sent into the operation space, the algorithm degenerates into DARTS. Considering the GPU memory's limitation, the search cannot be performed on 20 cells. To solve this problem, we introduce a progressive operation pruning strategy to the process of partial channel connections, which is inspired by P-DARTS.

In DARTS, there are eight candidate operations in total, and the importance of each operation differs. If we can constantly remove the worst candidate operations in the search process, we can greatly reduce the consumed GPU memory.

In the progressive partial channel connections strategy, we divide the algorithm search process into n stages. To reduce GPU memory consumption, in stage t, we sample $1/k_t$ of all channels for partial channel connections and retain the first l_t candidate operations with the highest importance. In stage $t+1$, we sample $1/k_{t+1}$ $(k_{t+1} > k_t)$ of all channels for partial channel connections and retain the first l_{t+1} $(l_t > l_{t+1})$ candidate operations with the highest importance. During the search process, we continuously reduce the number of candidate operations, which enables PA-DARTS to keep GPU memory consumption constant throughout the search process. Our entire search process can be carried out on 20 cells and can run on a regular 11G GPU.

P-DARTS proposes a progressive candidate operation pruning strategy to eliminate the large gaps between the architecture depths in the search stage and the validation stages. This simple method has been adopted by many subsequent methods. In PA-DARTS, we also adopt a progressive candidate operation pruning strategy to ensure that the entire search process can always be carried out on 20 cells. However, our pruning operation's operating mechanism differs from P-DARTS. In the early stage of the search, we only select a small number of channels for partial channel connections. At this time, the weight-free operations do not have big unfair competitive advantages over the convolution operations. We can safely prune some of the worst candidate operations. As the search continues, although the chosen number of channels increases, the weight-equipped operations are constantly being trained, and the unfair competitive advantages of the weight-free operations subsequently decrease. This allows us to gradually prune the unpromising candidate operations. For this reason, we always maintain fair competition among the candidate operations during our pruning process, so PA-DARTS does not have the problem of skip connection aggregation. In P-DARTS, the skip connections have unfair competitive advantages at the beginning, which leads to the pruning of a large number of convolution operations in the early stage of the search. Finally, the neural network architecture found by P-DARTS often has more than five skip connections, and the skip connections need to be manually replaced with convolution operations.

IV. Experiments

To test the performance of our algorithm, we conducted experiments on three commonly used datasets (CIFAR-10, CIFAR-100 and ImageNet) to verify the classification performance of

the network searched by our algorithm. We also transferred the searched network on CIFAR-10 to ImageNet for precision measurement. In addition, we designed ablation experiments to verify the effectiveness of the algorithm.

A. Datasets

To verify the effectiveness of PA-DARTS, we conducted experiments on three widely used datasets, namely CIFAR-10, CIFAR-100 and ImageNet. CIFAR-10 is a small dataset used to identify universal objects. A total of 10 categories of RGB pictures are included in the dataset with 50000 training pictures and 10000 test pictures. The training pictures were used to search for the network architecture, and the test pictures were used to verify the performance. Similar to CIFAR-10, CIFAR-100 has 100 categories, each of which contains 600 images, including 500 training images and 100 test images. ImageNet is currently the largest image recognition dataset in the world. The ISLVRC2012 training set has 1281167 photos, which are divided into 1000 categories. The test set of ISLVRC2012 has 100000 pictures.

B. Implementation Details

We used the DARTS search space, which is composed of normal cells and reduction cells. The reduction cells are located at the 1/3 and 2/3 positions of the whole network, and the rest are normal cells. Each cell consists of two input nodes, one output node and four intermediate nodes. Eight candidate operations are found between two adjacent nodes, which are *none*, *max_pool_3x3*, *avg_pool_3x3*, *skip_connect*, *sep_conv_5x5*, *sep_conv_3x3*, *dil_conv_3x3* and *dil_conv_5x5*. Thanks to our strategy, the search process and verification process were all conducted on 20 cells.

The architecture search process was divided into three stages. In the first stage, 1/8 of all channels were selected to send into the operation space for partial channel connections based on channel attention, and all eight candidate operations were reserved. In the second stage, 1/4 of all channels were selected, and five candidate operations were reserved. In the third stage, we selected all channels to be placed in the operation space, and only three candidate operations were reserved. At each stage, we trained for 20 epochs. We used the SGD optimiser to update the network parameter ω, and the initial learning rate was set to 0.025. We adjusted the learning rate using the cosine annealing strategy. We used the Adam optimiser to update the architecture parameter α. The initial learning rate was set to 0.0006.

C. Experimental Results and Analysis

1) Evaluation: We stacked 20 cells and trained the network for 600 epochs with a batch size of 64 during network accuracy verification. The SGD optimiser was used, and the initial learning rate was set to 0.025. To avoid overfitting, the cutout length was set to 16 for image enhancement. In addition, we used auxiliary loss, and the auxiliary weight was set to 0.4. Table I shows the results of the architecture searched by our algorithm on CIFAR-10 and CIFAR-100. We compared our algorithm with other state-of-the-art (SOTA) models. The top block represents the results from training the best-searched model, and the bottom block contains the average results obtained by searching multiple times. '$*$' denotes the architecture searched without weight inheritance.

The average precision of P-DARTS is obtained from β-DARTS [35]. The optimal architecture searched by PA-DARTS had the highest test classification accuracy of 97.59% on CIFAR-10 and 83.61% on CIFAR-100. To ensure the reliability of PA-DARTS, we carried out multiple experiments on CIFAR-10, and the average precision of the architecture searched was 97.52% $\pm$ 0.07. It can be seen in Table I that the classification accuracy of the neural network architectures searched by PA-DARTS was higher than that of manually designed neural network architectures, and the classification accuracy was also higher than that of the architectures obtained through automated searches using recently proposed methods. For instance, the classification accuracy of the neural network architectures for which we searched significantly exceeded that of manually designed networks, such as ResNet and DenseNet on CIFAR-10. Compared with the neural architecture search methods based on EC and RL, such as MFENAS and NPENAS, PA-DARTS not only had slightly superior accuracy but also advantages in search time. MSNAS had achieved advantages in search time, but its classification accuracy was lower than our algorithm. Compared with the original DARTS, which achieved a classification accuracy of 97% on CIFAR-10, PA-DARTS exhibited excellent performance. In addition, compared with recent studies on DARTS, such as β-DARTS, and CDARTS, PA-DARTS also achieved slight advantages. FP-DARTS achieved similar classification accuracy to ours, and its search time was shorter.

We transferred the network architecture searched on CIFAR-10 to ImageNet for evaluation, and the results are shown in Table II. We trained the network on the ImageNet dataset for 250 epochs with a batch size of 256. The initial channels were set to 48, and the initial learning rate was 0.1. The precision of the searched architecture reached the classification accuracy of 75.3% on ImageNet. It can be seen in Table II that PA-DARTS surpassed some recent SOTA algorithms.

2) Analysis of Searched Cells: Fig. 3 shows the normal cell and reduction cell searched by PA-DARTS. The architecture for which we searched only had two skip connections in normal cell and achieved outstanding performance, which was mainly due to the fact that our algorithm directly searched on the 20 cells without the problem of skip connection aggregation. In P-DARTS, due to the unfair competition between operations, skip connections dominate the search results, as there are more than four skip connections in normal cells. We conducted experiments on P-DARTS and found acute skip connection aggregation. P-DARTS can only manually select a fixed number of skip connections with the largest weights. This subjective method is obviously not convincing. The progressive partial channel connections proposed in this article ensures that the number of skip connections is always within a reasonable range (1–3), effectively solving the problem of skip connection aggregation.

D. Ablation Studies

1) Analysis of Partial Channel Connections: Due to the unfair competition among candidate operations, DARTS is prone

TABLE I
CLASSIFICATION ACCURACY ON CIFAR-10 AND CIFAR-100

Architecture	Search Cost (GPU Days)	Params (M)	Classification Accuracy		Search Methods
			CIFAR-10 (%)	CIFAR-100 (%)	
DenseNet-BC [36]	-	25.6	96.54	82.82	Manual
MobileNetV2 [37]	-	2.2	96.74	80.80	Manual
ENAS [38]	0.5	4.6	97.11	80.57	RL
SMASH [20]	1.5	16	95.97	-	RL
NASNet-A [39]	2000	3.3	97.35	83.18	RL
Genetic CNN [40]	17	-	92.90	70.95	Evo
AmoebaNet-B [41]	3150	2.8	97.45	-	Evo
CARTS-I [42]	0.4	3.6	97.38	-	Evo
EPCNAS-C [43]	1.10	1.16	96.93	-	Evo
NSGA-Net [44]	4	3.3	97.25	-	Evo
MFENAS [45]	0.6	2.94	97.41	-	Evo
DARTS(1st) [21]	0.4	3.4	97.00	82.24	GD
DARTS(2st) [21]	1	3.3	97.24	82.46	GD
SNAS [17]	1.5	2.8	97.15	82.45	GD
GDAS [28]	0.2	3.4	97.07	81.62	GD
P-DARTS [23]	0.3	3.4	97.50	82.51	GD
FairDARTS [25]	0.4	2.8	97.46	82.39	GD
PC-DARTS [22]	0.1	3.6	97.43	83.10	GD
RF-DARTS [46]	-	4.6	97.40	83.50	GD
EoiNAS [47]	0.6	3.4	97.50	82.70	GD
PA-DARTS	0.36	3.75	97.59	83.62	GD
PA-DARTS*	0.36	3.3	97.42	83.03	GD
NPENAS [48]	1.8	2.54	97.46 ± 0.10	-	Evo
MSNAS [49]	0.23	3.25	97.32 ± 0.08	-	Evo
P-DARTS [23]	0.3	3.3 ± 0.21	97.19 ± 0.14	-	GD
R-DARTS [50]	1.6	-	97.05 ± 0.21	81.99 ± 0.26	GD
DARTS+PT [51]	0.8	3.0	97.39 ± 0.08	-	GD
DARTS+AVD [52]	1.3	3.3	97.39 ± 0.02	-	GD
DARTS- [53]	0.4	3.5 ± 0.13	97.41 ± 0.08	82.49 ± 0.25	GD
βDARTS [35]	0.4	3.78 ± 0.08	97.49 ± 0.07	83.48 ± 0.03	GD
AGNAS [32]	0.4	3.6	97.47 ± 0.003	-	GD
FP-DARTS [54]	0.08	3.4	97.5 ± 0.05	83.50 ± 0.05	GD
CDARTS [55]	0.3	3.9 ± 0.08	97.52 ± 0.04	84.31	GD
PA-DARTS	0.36	3.57 ± 0.16	97.52 ± 0.07	83.46 ± 0.40	GD

The results above the double line are the best ones achieved by each method, while the results below the double line are the average ones over multiple experiments. '*' denotes that the architecture is searched without weight inheritance.

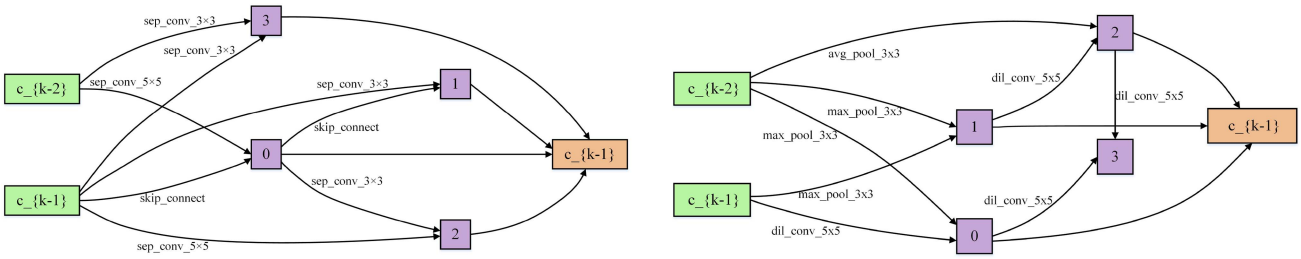


Fig. 3. Architectures searched by PA-DARTS on CIFAR-10. The picture on the left is the normal cell, and the picture on the right is the reduction cell.

to the phenomenon of skip connection aggregation. The partial channel connections mechanism based on the attention mechanism proposed here can solve this problem. To verify the effectiveness of this strategy, we conducted three different architecture searching experiments. Each experiment was divided into three phases. We used $[1/8, 1/4, 1]$ to represent that 1/8 of all channels were selected in the first stage, 1/4 of all channels were selected in the second stage, and all channels were selected in the third stage, In these three experiments, we set the number of channels selected in the three stages to $[1/8, 1/4, 1]$, $[1/4, 1/2, 1]$ and $[1, 1, 1]$, respectively. In the third experiment, we did not use partial channel connections.

Considering the limitations of GPU memory, all experiments were conducted on eight-layer cells. The experimental results are shown in Table III. As the number of selected channels increased, the number of weight-free operations in the network increased, and the network could not fully extract information from the pictures, resulting in a continuous decline in accuracy. In fact, if we had searched on a supernet with 20 cells with channels [1/8, 1/4, 1/2], there would still be two skip connections in the normal cells. We believe it is reasonable to have two skip connections in the network because not only can this prevent gradient disappearance but also ensures that the network can extract enough information.

TABLE II
CLASSIFICATION ACCURACY ON IMAGENET

Architecture	Search Cost (GPU Days)	Params (M)	Top 1 (%)	Top 5 (%)
NASNet-A [39]	1800	5.3	74.0	-
BNAS [56]	0.2	3.9	74.3	91.5
SPOS [57]	12	7.1	74.8	-
FairNAS [58]	12	4.4	74.7	-
CARS-I [59]	0.4	5.1	75.2	91.5
MFENAS [45]	0.6	5.98	73.94	91.82
BayesNAS [60]	4	3.9	72.7	91.2
DARTS [24]	4	4.9	73.1	-
GDAS [28]	0.2	5.3	74.0	-
SENAS-E [42]	8	5.2	74.72	-
PC-DARTS [22]	0.1	5.3	74.9	92.2
DARTS+PT [51]	0.8	4.6	74.5	92.0
SDARTS+ADV [61]	1.3	5.4	74.8	92.2
FairDARTS [25]	0.4	4.8	75.1	92.5
EoiNAS [47]	0.6	6.0	74.4	91.7
PA-DARTS	0.4	5.2	75.3	92.25

The horizontal line separates DARTS-based methods from other NAS methods.

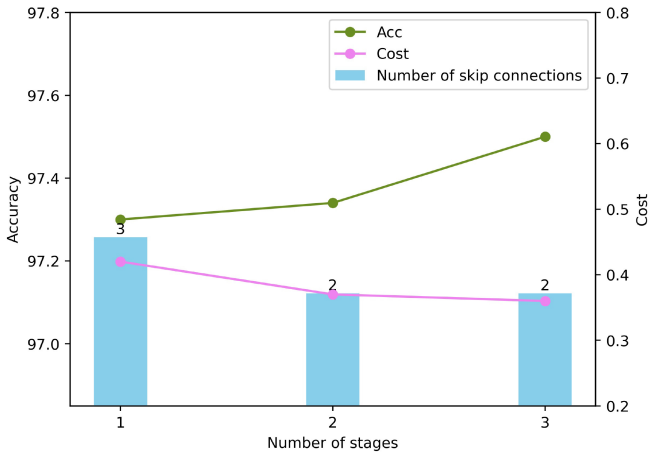


Fig. 4. Progressive search with different numbers of stages.

TABLE III
CLASSIFICATION ACCURACY WHEN SELECTING DIFFERENT CHANNEL NUMBERS

Number of Channels Selected	Number of Skip Connections	Accuracy
$[1/8, 1/4, 1]$	2	97.25
$[1/4, 1/2, 1]$	4	97.19
$[1, 1, 1]$	5	97.03

TABLE IV
ABLATION STUDIES ON CIFAR-10

Methods	Params (M)	Cost (GPU Days)	Accuracy
DARTS+partial connections	3.16	0.42	97.3
DARTS+progressive pruning	1.94	0.3	96.73
DARTS+progressive pruning*	3.4	0.3	97.5
PA-DARTS	3.75	0.36	97.59

2) Analysis of the Progressive Search: Our proposed progressive search strategy alleviated the instability of the partial channel connections by gradually increasing the number of selected channels in the partial channel connections. At the same time, the progressive pruning operation ensured that our entire search process was always carried out on 20 cells. To verify the effectiveness of the progressive search strategy, we designed three independent experiments in which we selected different numbers of search stages for the progressive search. Specifically, in the first experiment, we did not use the progressive search strategy and selected 1/4 channels for partial channel connections throughout the entire search process. In the second experiment, we used a two-stage search strategy. The number of channels in the two stages was set to [1/4, 1], and the number of candidate operations retained was set to 8 and 4. In the third experiment, we used a three-stage search strategy. The setup for the third experiment was the same as the experimental details in Part B.

In Fig. 4, it can be seen that the three-stage progressive search achieved the best classification accuracy because the three-stage search process balanced the unfair competition and solved the unstable training problems. The network architecture classification accuracy obtained from the one-stage search was the lowest, and we found that the one-stage search results were unstable in this experiment.

To further validate the effectiveness of the progressive search, we conducted ablation studies on the partial channel connections and pruning operations. Table IV provides the experimental results, where '*' denotes that the number of skip connections had been manually filtered. Clearly, our proposed progressive search strategy achieved the highest classification accuracy. When only pruning operations were used, we found that the searched network architecture often contained more than five skip connections, which seriously affected the performance of the network architecture.

3) Analysis of Channel Attention: We used the channel attention mechanism to select more important channels for partial channel connections, which alleviated the search instability caused by the information loss. We designed ablation experiments to verify the performance of the channel attention mechanism in the architecture search process. In the ablation experiment, we did not use the channel attention mechanism, nor did we use the edge regularisation strategy proposed in PC-DARTS. The classification accuracy of the final searched architecture on CIFAR-10 was 97.21. Undoubtedly, the attention mechanism greatly improved the performance of the algorithm. This was because, in the first two stages of the search, we only selected a small number of channels for partial connections, such as 1/8 of all channels in the first stage. If we had randomly selected channels at this point, it would have caused a large amount of information loss and affected the stability of the supernet training. In the experiment, we found that if the attention mechanism is not applied, the convolutional operations and skip connections will be pruned in the first stage of the search, while a large number of pooling operations will be retained, thus affecting the subsequent searches.

4) Analysis of the Number of Stacking Layers: To verify the impact of the number of cells stacked in the search stage, we conducted experiments on supernets with 8 cells, 14 cells and 20 cells and evaluated the performance of the neural architecture we finally found by stacking 20 cells. In all three groups of experiments, we applied the progressive partial channel connections based on the attention mechanism, and the number of

TABLE V
CLASSIFICATION ACCURACY WHEN STACKING DIFFERENT LAYERS

Number of cells	GPU (Days)	Accuracy
8	0.15	97.25
14	0.25	97.39
20	0.36	97.59

channels selected was set to $[1/8, 1/4, 1]$. Table V shows the final experimental results.

We found that the closer the number of training cells was to the number of verification cells, the higher the final classification accuracy of the verification set. This was a very intuitive phenomenon. A cell that performs well in a shallow network may not perform well in a deep network. Our algorithm solved the depth gap between the search stage and the verification stage. Although we increased the search time minimally, we made great progress in classification accuracy.

5) Analysis of Weight Inheritance: In the process of progressive partial channel connections, we used the parameter inheritance strategy to ensure that the weight ω at stage t was trained at stage $t-1$ to reduce the unfair competition between the weight-free and weight-equipped operations. However, the number of channels in the later stage was usually more than that in the previous stage. Thus, each stage could only partially inherit the weight information of the previous stage. We found that this partial weight inheritance was effective through the experiments. In the second stage, we inherited the weight of the first stage and achieved 56% accuracy with only one epoch of training, while in the first stage, we used seven epochs to achieve the same accuracy.

To further verify the effectiveness of weight inheritance, we conducted ablation experiments without weight inheritance. All the settings in the search process were the same as before, except that the weight inheritance strategy was not used. The architecture for which we searched achieved a classification accuracy of 97.42% on CIFAR-10 with 3.3 M parameters. Although the precision was slightly reduced compared with the algorithm using weight inheritance, it could still be maintained at a high level. We attributed this phenomenon to two reasons. First, the whole network was still searching on 20 cells, and there was no depth gap. Second, progressive pruning could prune some very unpromising weight-free operations during the search process, which prevented the aggregation of skip connections to some extent.

V. CONCLUSIONS AND FUTURE WORK

In this article, PA-DARTS was proposed to solve the problems of the depth gap between the training phase and the validation phase as well as alleviate skip connection aggregation. In PA-DARTS, we only selected a few key channels for convolutional operations and reserved all candidate operations in the early stage of the search. As the search progressed, we gradually increased the number of channels selected and pruned the unpromising candidate operations. Our algorithm increased the stability of the search process and allowed the search phase and the verification phase to be carried out on all 20 cells. We verified our algorithm on the CIFAR-10, CIFAR-100 and ImageNet datasets, and experiments showed that PA-DARTS was stable and superior. In future work, we will optimise our search space and try to reduce the cost time of our algorithm. We believe that the search space of DARTS is redundant, and it is meaningful to simplify the supernet. In addition, we are considering introducing different channel attention mechanisms to improve the performance of the algorithm.

REFERENCES

[1] L. Chen et al., "Deep neural network based vehicle and pedestrian detection for autonomous driving: A survey," *IEEE Trans. Intell. Transp. Syst.*, vol. 22, no. 06, pp. 3234–3246, Jun. 2021.

[2] A. Dudhane, P. W. Patil, and S. Murala, "An end-to-end network for image de-hazing and beyond," *IEEE Trans. Emerg. Topics Comput. Intell.*, vol. 6, no. 01, pp. 159–170, Feb. 2022.

[3] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *Proc. Int. Conf. Learn. Representations*, 2015, pp. 1–14.

[4] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2016, pp. 770–778.

[5] C. Szegedy et al., "Going deeper with convolutions," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2015, pp. 1–9.

[6] B. Lyu, S. Wen, K. Shi, and T. Huang, "Multiobjective reinforcement learning-based neural architecture search for efficient portrait parsing," *IEEE Trans. Cybern.*, vol. 53, no. 2, pp. 1158–1169, Feb. 2023.

[7] Y. Liu, Y. Sun, B. Xue, M. Zhang, G. G. Yen, and K. C. Tan, "A survey on evolutionary neural architecture search," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 34, no. 02, pp. 550–570, Feb. 2023.

[8] B. Zoph and Q. Le, "Neural architecture search with reinforcement learning," in *Proc. Int. Conf. Learn. Representations*, 2017, pp. 1–16.

[9] H. Zhu and Y. Jin, "Real-time federated evolutionary neural architecture search," *IEEE Trans. Evol. Computation*, vol. 26, no. 02, pp. 364–378, Apr. 2022.

[10] D. Szwarcman, D. Civitarese, and M. Vellasco, "Quantum-inspired evolutionary algorithm applied to neural architecture search," *Appl. Soft Comput.*, vol. 120, 2022, Art. no. 108674.

[11] Y. Sun, B. Xue, M. Zhang, and G. G. Yen, "Completely automated CNN architecture design based on blocks," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 31, no. 04, pp. 1242–1254, Apr. 2020.

[12] Y. Xue, P. Jiang, F. Neri, and J. Liang, "A multi-objective evolutionary approach based on graph-in-graph for neural architecture search of convolutional neural networks," *Int. J. Neural Syst.*, vol. 31, no. 09, 2021, Art. no. 2150035.

[13] E. Real, A. Aggarwal, Y. Huang, and Q. V. Le, "Aging evolution for image classifier architecture search," in *Proc. AAAI Conf. Artif. Intell.*, 2019, pp. 5048–5056.

[14] Y. Chen, R. Gao, F. Liu, and D. Zhao, "ModuleNet: Knowledge-inherited neural architecture search," *IEEE Trans. Cybern.*, vol. 52, no. 11, pp. 11661–11671, Nov. 2022.

[15] D. O'Neill, B. Xue, and M. Zhang, "Evolutionary neural architecture search for high-dimensional skip-connection structures on densenet style networks," *IEEE Trans. Evol. Comput.*, vol. 25, no. 6, pp. 1118–1132, Dec. 2021.

[16] A. Klein, S. Falkner, J. T. Springenberg, and F. Hutter, "Learning curve prediction with Bayesian neural networks," in *Proc. Int. Conf. Learn. Representations*, 2016, pp. 1–16.

[17] S. Xie, H. Zheng, C. Liu, and L. Lin, "SNAS: Stochastic neural architecturesearch," in *Proc. Int. Conf. Learn. Representations*, 2019, pp. 1–17.

[18] H. Zhang, Y. Jin, and K. Hao, "Evolutionary search for complete neural network architectures with partial weight sharing," *IEEE Trans. Evol. Comput.*, vol. 26, no. 5, pp. 1072–1086, Oct. 2022.

[19] Y. Tang et al., "A semi-supervised assessor of neural architectures," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2020, pp. 1810–1819.

[20] A. Brock, T. Lim, J. M. Ritchie, and N. Weston, "SMASH: one-shot model architecture search through hypernetworks," in *Proc. Int. Conf. Learn. Representations*, 2018, pp. 1–21.

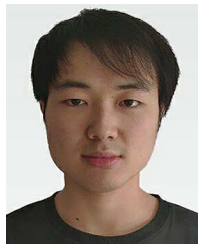
[21] H. Liu, K. Simonyan, and Y. Yang, "DARTS: Differentiable architecture-search," in *Proc. Int. Conf. Learn. Representations*, 2019, pp. 1–13.

[22] Y. Xu et al., "PC-DARTS: Partial channel connections for memory-efficient architecture search," in *Proc. Int. Conf. Learn. Representations*, 2020, pp. 1–13.
[23] X. Chen, L. Xie, J. Wu, and Q. Tian, "Progressive differentiable architecture search: Bridging the depth gap between search and evaluation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 1294–1303.
[24] H. Liang, S. Zhang, J. Sun, X. He, W. Huang, K. Zhuang, and Z. Li, "DARTS+ : Improved differentiable architecture search with early stopping,"2019. *arXiv:1909.06035*.
[25] X. Chu, T. Zhou, B. Zhang, and J. Li, "Fair DARTS: Eliminating unfair advantages in differentiable architecture search," in *Proc. Eur. Conf. Comput. Vis.*, 2020, pp. 465–480.
[26] P. Zhou, C. Xiong, R. Socher, and S. C. H. Hoi, "Theory-inspired path-regularized differential network architecture search," *Adv. Neural Inf. Process. Syst.*, pp. 8296–8307, vol. 33, 2020.
[27] G. Li, X. Zhang, Z. Wang, Z. Li, and T. Zhang, "StacNAS: Towards stable and consistent optimization for differentiable Neural Architecture Search," 2019. [Online]. Available: https://openreview.net/pdf?id=rygpAnEKDH
[28] X. Dong and Y. Yang, "Searching for a robust neural architecture in four GPU hours," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 1761–1770.
[29] J. Hu, L. Shen, and G. Sun, "Squeeze-and-excitation networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 7132–7141.
[30] S. Woo, J. Park, J.-Y. Lee, and I. S. Kweon, "CBAM: Convolutional block attention module," in *Proc. Eur. Conf. Comput. Vis.*, 2018, pp. 3–19.
[31] Z. Liu, H. Wang, S. Zhang, G. Wang, and J. Qi, "NAS-SCAM: Neural architecture search-based spatial and channel joint attention module for nuclei semantic segmentation and classification," in *Proc. Int. Conf. Med. Image Comput. Comput. Assist. Interv.*, 2020, pp. 263–272.
[32] Z. Sun et al., "AGNAS: Attention-guided micro and macro-architecture search," in *Proc. Int. Conf. Mach. Learn.*, 2022, pp. 20777–20789.
[33] Y. Xue and J. Qin, "Partial connection based on channel attention for differentiable neural architecture search," *IEEE Trans. Ind. Inform.*, vol. 19, no. 5, pp. 6804–6813, May 2023.
[34] Z. Wang, W. Zhang, and Z. Wang, "G-DARTS-A: Groups of channel parallel sampling with attention," 2020. *arXiv:2010.08360*.
[35] P. Ye, B. Li, Y. Li, T. Chen, J. Fan, and W. Ouyang, "βdarts: Beta − decay regularization for differentiable architecture search," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2022, pp. 10874–10883.
[36] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 4700–4708.
[37] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "Mobilenetv2: Inverted residuals and linear bottlenecks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 4510–4520.
[38] H. Pham, M. Guan, B. Zoph, Q. Le, and J. Dean, "Efficient neural architecture search via parameters sharing," in *Proc. Int. Conf. Mach. Learn.*, 2018, pp. 4095–4104.
[39] B. Zoph, V. Vasudevan, J. Shlens, and Q. V. Le, "Learning transferable architectures for scalable image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 8697–8710.
[40] L. Xie and A. Yuille, "Genetic CNN," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 1379–1388.
[41] E. Real, A. Aggarwal, Y. Huang, and Q. V. Le, "Regularized evolution for image classifier architecture search," in *Proc. AAAI Conf. Artif. Intell.*, 2019, pp. 4780–4789.
[42] Y. Hu, X. Wang, L. Li, and Q. Gu, "Improving one-shot NAS with shrinking-and-expanding supernet," *Pattern Recognit.*, vol. 118, 2021, Art. no. 108025.
[43] J. Huang, B. Xue, Y. Sun, M. Zhang, and G. G. Yen, "Particle swarm opti-mization for compact neural architecture search for image classification," *IEEE Trans. Evol. Comput* ., early access, May 24, 2023, doi: 10.1109/TEVC.2022.3217290.
[44] Z. Lu et al., "NSGA-Net: Neural architecture search using multi-objective genetic algorithm," in *Proc. Genet. Evol. Comput. Conf.*, 2019, pp. 419–427.
[45] S. Yang, Y. Tian, X. Xiang, S. Peng, and X. Zhang, "Accelerating evolutionary neural architecture search via multifidelity evaluation," *IEEE Trans. Cogn. Develop. Syst.*, vol. 14, no. 4, pp. 1778–1792, Dec. 2022.
[46] Zhang, Y. Li, X. Zhang, Y. Wang, and J. Sun, "Differentiable architec-ture search with random features," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2023, pp. 16060–16069.
[47] Y. Zhou, X. Xie, and S.-Y. Kung, "Exploiting operation importance for differentiable neural architecture search," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 33, no. 11, pp. 6235–6248, Nov. 2022.
[48] C. Wei, C. Niu, Y. Tang, Y. Wang, H. Hu, and J. Liang, "NPENAS: Neural predictor guided evolution for neural architecture search," *IEEE Trans. Neural Netw. Learn. Syst.*, early access, Mar. 01, 2022, doi: 10.1109/TNNLS.2022.3151160.
[49] Dong, B. Hou, L. Feng, H. Tang, K. C. Tan, and Y. Ong, "A cell-based fast memetic algorithm for automated convolutional neural architecture design," *IEEE Trans. Neural Netw. Learn. Syst.*, early access, Mar. 17, 2022, doi: 10.1109/TNNLS.2022.3155230.
[50] A. Zela, T. Elsken, T. Saikia, Y. Marrakchi, T. Brox, and F. Hutter, "Understanding and robustifying differentiable architecture search," in *Proc. Int. Conf. Learn. Representations*, 2020, pp. 1–15.
[51] R. Wang, M. Cheng, X. Chen, X. Tang, and C.-J. Hsieh, "Rethinking architecture selection in differentiable NAS," in *Proc. Int. Conf. Learn. Representations*, 2021, pp. 1–18.
[52] X. Chen and C.-J. Hsieh, "Stabilizing differentiable architecture search via perturbation-based regularization," in *Proc. Int. Conf. Mach. Learn.*, 2020, pp. 1554–1565.
[53] X. Chu, X. Wang, B. Zhang, S. Lu, X. Wei, and J. Yan, "DARTS-: Robustly stepping out of performance collapse without indicators," in *Proc. Int. Conf. Learn. Representations*, 2020, pp. 1–22.
[54] W. Wang, X. Zhang, H. Cui, H. Yin, and Y. Zhang, "FP-DARTS: Fast parallel differentiable neural architecture search for image classification," *Pattern Recognit.*, vol. 136, 2023, Art. no. 109193.
[55] H. Yu et al., "Cyclic differentiable architecture search," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 45, no. 1, pp. 211–228, Jan. 2023.
[56] Z. Ding, Y. Chen, N. Li, D. Zhao, Z. Sun, and C. P. Chen, "BNAS: Efficient neural architecture search using broad scalable architecture," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 33, no. 9, pp. 5004–5018, Sep. 2022.
[57] Z. Guo et al., "Single path one-shot neural architecture search with uniform sampling," in *Proc. Eur. Conf. Comput. Vis.*, 2020, pp. 554–560.
[58] X. Chu, B. Zhang, and R. Xu, "FairNAS: Rethinking evaluation fairness of weight sharing neural architecture search," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2021, pp. 12239–12248.
[59] Z. Yang et al., "CARS: Continuous evolution for efficient neural architecture search," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2020, pp. 1829–1838.
[60] H. Zhou, M. Yang, J. Wang, and W. Pan, "Bayesnas: A Bayesian approach for neural architecture search," in *Proc. Int. Conf. Mach. Learn.*, 2019, pp. 7603–7613.
[61] X. Chen and C.-J. Hsieh, "Stabilizing differentiable architecture search via perturbation-based regularization," in *Proc. Int. Conf. Mach. Learn.*, 2020, pp. 1554–1565.

Yu Xue (Member, IEEE) received the Ph.D. degree in application technology of computer from the School of Computer Science and Technology, Nanjing University of Aeronautics and Astronautics, Nanjing, China, in 2013. From 2016 to 2017, he was a Visiting Scholar with the School of Engineering and Computer Science, Victoria University of Wellington, Wellington, New Zealand. From 2017 to 2018, he was a Research Scholar with the Department of Computer Science and Engineering, Michigan State University, East Lansing, MI, USA. His research interests include deep learning, evolutionary computation, machine learning, and computer vision.

Changchang Lu received the bachelor's degree in computational mathematics from the School of Science, North China University of Science and Technology, Tangshan, China, in 2021. He is currently working toward the master's degree in software engineering with the Nanjing University of Information Science and Technology, Nanjing, China. His research interests include deep learning and neural architecture search.

Ferrante Neri (Senior Member, IEEE) received the Laurea and Ph.D. degrees in electrical engineering from the Technical University of Bari, Bari, Italy, in 2002 and 2007, respectively, and the second Ph.D. degree in scientific computing and optimization and the D.Sc. degree in computational intelligence from the University of Jyväskylä, Jyväskylä, Finland, in 2007 and 2010, respectively. Between 2009 and 2014, he was an Academy Research Fellow with the Academy of Finland to lead the project Algorithmic Design Issues in Memetic Computing. He was with De Montfort University, Leicester, U.K., between 2012 and 2019 and with the University of Nottingham, Nottingham, U.K., between 2019 and 2022. Since 2022, he has been with the University of Surrey, Guildford, as a Full Professor of machine learning and artificial intelligence and the Head of the Nature Inspired Computing and Engineering (NICE) Research Group. His research focuses on metaheuristic optimisation with applications in the context of machine learning.



Jiafeng Qin received the bachelor's and master's degrees in software engineering from the Nanjing University of Information Science and Technology, Nanjing, China, in 2020 and 2023, respectively. His research interests include deep learning, evolutionary computation, and neural architecture search.