An Effective One-Shot Neural Architecture Search Method with Supernet Fine-Tuning for Image Classification

Gonglin Yuan, Bing Xue, and Mengjie Zhang
School of Engineering and Computer Science, Victoria University of Wellington
Wellington, New Zealand
{gonglin.yuan,bing.xue,mengjie.zhang}@ecs.vuw.ac.nz

ABSTRACT

Neural architecture search (NAS) is becoming increasingly popular for its ability to automatically search for an appropriate network architecture, avoiding laborious manual designing processes, and potentially introducing novel structures. However, many NAS methods suffer from heavy computational consumption. One-shot NAS alleviates this issue by training a big supernet and allowing all the candidates to inherit weights from the supernet, avoiding training from scratch. However, the performance evaluations in one-shot methods might not always be reliable due to the weight co-adaption issue inside the supernet. This paper proposes a supernet fine-tuning strategy to allow the supernet's weights to adapt to the new focused search region along with the evolutionary process. Furthermore, a new genetic algorithm-based search method is designed to offer an effective path-sampling strategy in the search region and provide a new population generation method to preclude unfair fitness comparisons between different populations. The experimental results demonstrate the proposed method achieves promising results compared with 32 peer competitors in terms of the algorithm's computational cost and the searched architecture's performance. Specifically, the proposed method achieves classification error rates of 2.50% and 17.07% within only 0.50 and 0.92 GPU-days on CIFAR10 and CIFAR100, respectively.

CCS CONCEPTS

• **Computing methodologies** → **Search methodologies**; **Computer vision**; **Machine learning**.

KEYWORDS

neural architecture search, evolutionary computation, image classification

ACM Reference Format:
Gonglin Yuan, Bing Xue, and Mengjie Zhang. 2023. An Effective One-Shot Neural Architecture Search Method with Supernet Fine-Tuning for Image Classification. In *Genetic and Evolutionary Computation Conference (GECCO '23), July 15–19, 2023, Lisbon, Portugal.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3583131.3590438

GECCO '23, July 15–19, 2023, Lisbon, Portugal

ACM ISBN 979-8-4007-0119-1/23/07…$15.00
https://doi.org/10.1145/3583131.3590438

1 INTRODUCTION

Neural Architecture Search (NAS) has become a popular research topic in recent years [17, 25], and many networks constructed by NAS have achieved state-of-the-art performance in various computer vision tasks. Compared with manually designing networks, NAS approaches can not only avoid the laborious and error-prone designing process, but may also break through the bottleneck of manual design to obtain effective novel network architectures.

In addition to the merits mentioned above, most NAS algorithms suffer from the prohibitive computational cost since each candidate network's performance needs to be evaluated during the search, which involves the training of the weights and the testing of the trained network. In an effort to mitigate the computational burden, researchers have employed shallow training [28], surrogate-assisted models [34], and parameter sharing [1] techniques [41]. Specifically, shallow training methods include training the network architecture for a small number of epochs, using a small portion of the training dataset, and training a partial network. However, shallow training may introduce inaccurate assessments that may undermine the search. On the other hand, surrogate-assisted models can help predict the performance of a network, but designing an accurate surrogate model is not easy, and the training of the surrogate may also consume computing resources. In contrast, one-shot NAS algorithms that employ the parameter-sharing technique have gained widespread favor for their high efficiency.

One-shot NAS algorithms only train the supernet once, and all candidate networks directly inherit weights from the supernet, avoiding training from scratch. Current one-shot NAS algorithms can be divided into gradient-based and sample-based methods [29]. Gradient-based one-shot approaches introduce architecture parameters that can be optimized jointly with the weights of the supernet [16]. However, such methods typically require high memory consumption. On the other hand, sample-based one-shot NAS methods consist of two consecutive stages: the training of the supernet and the search inside the supernet. In the first stage, an extensive network covering the entire search space (a.k.a. supernet) is predefined, and the weights of the supernet are trained by sampling various paths (a.k.a. subnets) within the supernet and updating the corresponding weights. In the second stage, a search method is employed to find the best network. While evaluating candidate networks, their weights are inherited from the supernet instead of being trained from scratch, significantly reducing the computational consumption.

[1] Some researchers use the word "weight sharing", but "weight sharing" is also used to represent the technique that can detect features at any location on the input, which is the foundation of convolutional neural networks [14, 26]. To avoid confusion, we used the word "parameter sharing" to indicate different networks use common parameters/weights from the supernet in this paper.

However, sample-based one-shot NAS methods also suffer from the shortcoming of inaccurate evaluations, i.e., the performance ranking correlation between weights inherited from the supernet and from stand-alone training may not be consistent [36, 39]. This is due to the weight co-adaption problem [3], which is caused by the large sharing extent of the supernet. Specifically, one-shot NAS is based on the assumption that all the subnets containing the same operation at the same layer can share identical weights of the operation. In practice, the weights of each operation are influenced by all the sampling paths that contain it. However, the same operation may need different weights in different paths/subnets. Thus, one-shot NAS methods rule out the optimized weights of different operations to adjust different subnets.

To alleviate the inferior evaluation problem of one-shot NAS, researchers have attempted to reduce the sharing extent of the supernet by training several copies of weights [29, 43], but these methods may introduce large memory consumption. Supposing the search space gets smaller and fewer subnets share the same operation's weights, the performance evaluated by the inherited weights will get closer to the stand-alone trained weights, and thus benefit the performance ranking, which inspires the idea of fine-tuning the supernet based on the gradually focused regions during the search process. This paper proposes combining the fine-tuning of the supernet and the search process: the search can guide the fine-tuning of the supernet, and the fine-tuned weights can provide more accurate performance evaluations for the search in turn.

The search is expected to gradually focus on smaller but more promising regions during the search process, thus guiding the supernet's fine-tuning; however, it is difficult to specify the refined search regions. This paper proposes a new Genetic Algorithm (GA)-based search strategy that can focus on shrunk refined search regions for each generation, while also offering an effective way to sample paths within the regions to support the supernet fine-tuning.

This paper aims to design an effective and efficient one-shot NAS algorithm that can automatically search for a promising network architecture, and the supernet is fine-tuned along with the search to provide more accurate performance evaluations of the subnets. The objectives are specified as follows:

1) Analyze the path sampling strategy of the supernet training, and provide a theoretical analysis on the impact of shrinking the focused regions to support the idea of supernet fine-tuning.
2) Propose a supernet fine-tuning method to provide more accurate performance estimations along with the search. The weights of the supernet are expected to fit the refined regions and offer more accurate assessments for the candidate subnets inside the region to guide the search.
3) Propose a new simple but effective GA-based search scheme to cope with the proposed supernet fine-tuning strategy. The search is expected to focus on smaller regions along with the evolution, and the new GA-based scheme can provide an effective path sampling strategy within the search regions.
4) Develop a new population generation method to avoid unfair comparisons between the current populations and the offspring populations since the populations may inherit weights from the supernets of different weights.

2 RELATED WORK

In one-shot NAS, the supernet is trained only once, and all the candidate networks can directly inherit weights from the supernet instead of being trained from scratch, substantially reducing the computational cost. In the supernet training stage, various paths are sampled and trained to update the weights. To make sure the subnets with the inherited weights perform consistently with stand-along training, researchers tried to apply effective sampling path methods for the supernet training: Guo *et al.* [9] employed a uniform path sampling strategy to alleviate the weight co-adaption problem; Chu *et al.* [7] proposed a fair sampling method that considers both the expectation fairness and strict fairness; You *et al.* [38] built a candidate path sampling pool to improve the training efficiency. However, Yu *et al.* [39] pointed out that the supernet weight inheritance strategy still degrades the candidates' ranking relationship and cannot reflect the candidate's actual performance.

Bender *et al.* [3] attributed the poor ranking correlation to the large sharing extent of the supernet, and researchers have tried to reduce the effect of weight co-adaption in the supernet. For example, instead of training only one supernet, Su *et al.* [29] prepared several supernets and combined their weights to evaluate candidate subnets. Zhao *et al.* [43] divided the search space into several regions and trained a corresponding supernet for each region to alleviate the undesired co-adaption. However, these methods introduce further training and increase memory consumption to save more supernets. Zhou *et al.* [45] proposed an adaptive curriculum learning method to gradually reduced the sharing extent during the training of the supernet, but the learning process is complicated and introduces a greater amount of computation.

In this paper, we reduce the sharing extent of the supernet in the search process. Specifically, the evolutionary process guides the fine-tuning of the supernet to provide more reliable evaluations. In this way, only a single copy of the supernet's weights needs to be saved, and no reducing sharing extent policy needs to be specifically designed or trained/optimized.

3 THE PROPOSED METHOD

This section will provide the framework and details of the proposed method. This effective one-shot method is based on supernet fine-tuning and GA, so we term it EOFGA.

3.1 Rethinking and Motivations

Generally, sample-based one-shot NAS approaches can be divided into two phases: supernet training and architecture search. In the first phase, a supernet that covers the whole search space $\mathcal{A}$ is trained by sampling different subnets (a.k.a. paths) from a uniform distributed space, i.e., $a \sim U(\mathcal{A})$. The supernet training follows:

$$W^*_{\mathcal{A}} = \arg\min_{W} \mathop{\mathbb{E}}_{a\sim U(\mathcal{A})} [\![\mathcal{L}_{tra}(a, W(a))]\!], \tag{1}$$

where W denotes the weights of the supernet, $\mathbb{E}[\![\cdot]\!]$ represents the expectation of variables, and $\mathcal{L}_{tra}(a, W(a))$ refers to the training loss of subnet a with weights $W(a)$ on the training data.

In the second stage, a subnet a inherits corresponding weights $W^*_{\mathcal{A}}(a)$ from the trained supernet, and the subnet with the highest

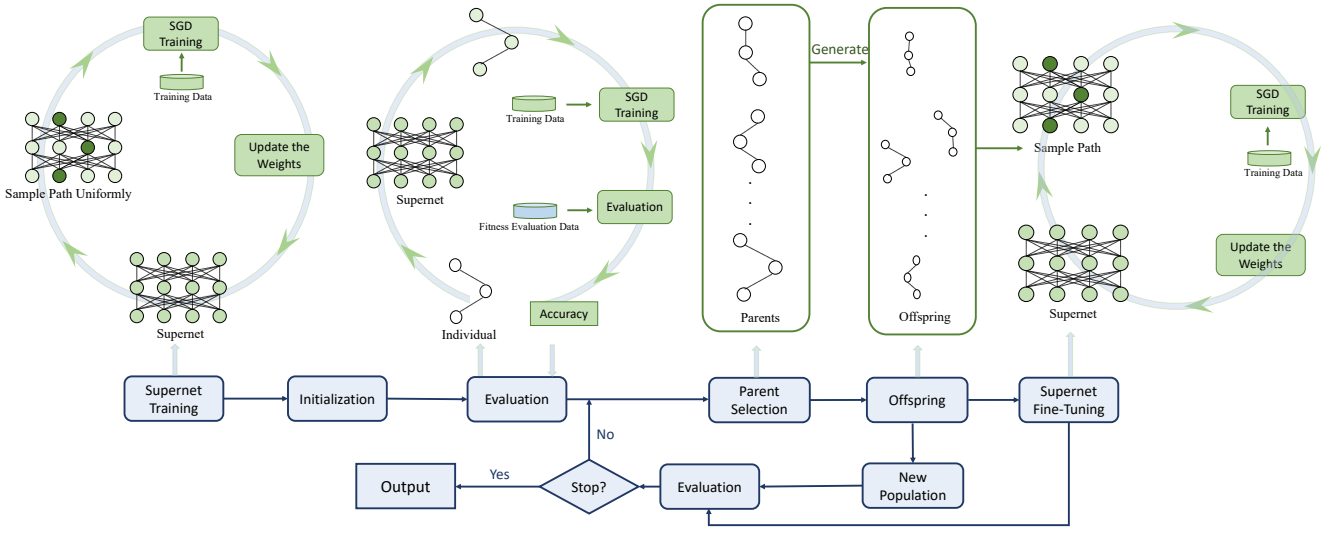

Figure 1: The overall framework of the proposed EOFGA.

accuracy on the evaluation data is expected to be searched, i.e.,

$$a^* = \arg\max_{a \in \mathcal{A}} \mathrm{ACC}_{eva}(a, \mathrm{W}^*_{\mathcal{A}}(a)) , \tag{2}$$

where ACC_{eva} is the accuracy of the candidate architecture on the evaluation data. It is evident that the evaluation accuracy of the candidate subnet is closely related to the inherited weights $W^*_{\mathcal{A}}(a)$, which is determined by the supernet training.

Suppose the supernet has L layers, and the number of optional operations for each layer is O. There will be O^L subnets within the supernet. For an operation o at a particular layer l, there are $O^{(L-1)}$ subnets containing the operation; thus, the weights of o^l are affected by a total number of $O^{(L-1)}$ subnets, which is typically a fairly large number. In theory, reducing the number of involved subnets can help the weights of o^l get close to the stand-alone training in a specific subnet, improving the evaluation quality. It is supported by [3], which suggests that reducing the sharing extent can improve the reliability of evaluations.

However, how to reduce the sharing extent is a problem. We think about this problem from the view of the function of the supernet — evaluating the candidate subnets in the search process. We can propose an evolutionary method to make the search gradually focus on smaller and more promissing regions from generation to generation, and the supernet can be fine-tuned according to the shrunk search region of each generation, providing more accurate evaluations for the individuals in the current shrunk region.

For the i-th generation, the supernet can be further fine-tuned according to the uniform distribution of the specific shrunk region $U(\mathcal{A}_i)$, i.e.,

$$W^*_{\mathcal{A}_i} = \arg\min_{W} \mathop{\mathbb{E}}_{a \sim U(\mathcal{A}_i)} [\![\mathcal{L}_{tra}(a, W(a))]\!] . \tag{3}$$

In this way, the sharing extent is reduced and the weight co-adaption problem is alleviated, thus improving the evaluation reliability. However, there are mainly two issues about the increased computation and the specification of the shrunken search region.

Specifically, the first issue is that fine-tuning the supernet introduces additional computational costs, but it will be quite a small amount, and the total computation cost is not significantly increased due to the following reasons.

1) The initial weights of the supernet for the i-th generation are inherited from the $(i-1)$-th generation, and the two focused search regions $\mathcal{A}_i$ and $\mathcal{A}_{i-1}$ are close to each other. Consequently, the difference between the optimized weights of the two consecutive generations $W^*_{\mathcal{A}_i}$ and $W^*_{\mathcal{A}_{i-1}}$ is usually small.
2) Typically, $\mathcal{A}_i$ is getting smaller along with the increase of i, and the training is much easier (a.k.a. the convergence of weights is faster) in a smaller region.

Another issue is regarding the specification of the new shrunken search region for each generation $\mathcal{A}_i$. The primary purpose of $\mathcal{A}_i$ is to guide the path sampling, i.e., $a \sim U(\mathcal{A}_i)$, for fine-tuning. Consequently, it is not essential to precisely specify a shrunken region for each generation. Instead, an effective sampling strategy within the new region is needed. This paper proposes a new strategy to sample candidates based on the selected parent individuals to construct a refined region in each generation.

3.2 Algorithm Overview

Figure 1 illustrates the main procedures of EOFGA, which is composed of the supernet training and a GA search pipeline, and particularly, the supernet is fine-tuned along with the evolution.

Before evolution, the supernet is predefined and trained for a predefined number of epochs. In one epoch, for each batch of data, a path/subnet within the supernet is selected, the Stochastic Gradient

Descent (SGD) training is applied, and then the corresponding weights are updated.

The population is then initialized and evaluated. Specifically, while evaluating an individual, the weights are inherited from the supernet and are then trained. The candidate network is evaluated on the fitness evaluation data, and the classification accuracy is used as the individual's fitness. A parent selection strategy is employed to select well-performing individuals as the parents, and the parents generate offspring based on the proposed crossover and mutation strategies. Afterwards, a supernet is fine-tuned. Specifically, the paths within the region of the offspring distribution are selected, and the weights are trained and updated. Simultaneously, a new population is generated. The individuals are evaluated following the similar evaluation method mentioned above. If the stopping criterion is not satisfied, a new parent population will be generated; otherwise, the best individual will be output as the searched network architecture.

3.3 Search Space and Individual Representation

In this work, we search for the best depth (the number of layers), width (the number of feature maps), and kernel sizes of convolutions. Figure 2 shows the general architecture of a candidate network and the optional operations. Following MobileNetV2 [27], a network comprises a stem, five consecutive cells, and a tail. The stem and the tail are all fixed and not in the search space. The stem is composed of a convolutional layer with a kernel size of 3×3. The tail consists of a convolutional layer and a fully-connected layer. There may be up to four blocks and at least one block within a single cell. Each block is composed of a 1×1 convolution to increase the number of feature maps, a depthwise separable convolution to extract informative features, and a 1×1 convolution to reduce the number of feature maps. The stride of the depthwise separable convolution within the first block in one cell may be one or two, depending on the size of the input image since the convolution with a stride of 2 will reduce the size of the feature map, but the feature map's size is at least 1×1; the other blocks in a cell are of the stride of one. The presence or absence of a block is searchable. Furthermore, within a block, the expansion rate of the first 1×1 convolution e, and the kernel size of the depthwise separable convolution k are both searchable, with 6 candidate expansion rates and 3 candidate kernel sizes:

- The expansion rate e: 1; 2; 3; 4; 5; 6;
- The kernel size $k \times k$: 3×3; 5×5; 7×7.

If one block exists, there are $6 \times 3 = 18$ options, and the possibilities of each block build the entire search space of EOFGA.

A candidate architecture representation is composed of five cell vectors. If the cell vector is [[1,3],[3,5],[5,5]], the cell consists of three blocks. The expansion rate and kernel size of the respective blocks are as follows: the first block has an expansion rate of 1 and a kernel size of 3×3; the second block has an expansion rate of 3 and a kernel size of 5×5; and the third block has an expansion rate of 5 and a kernel size of 5×5.

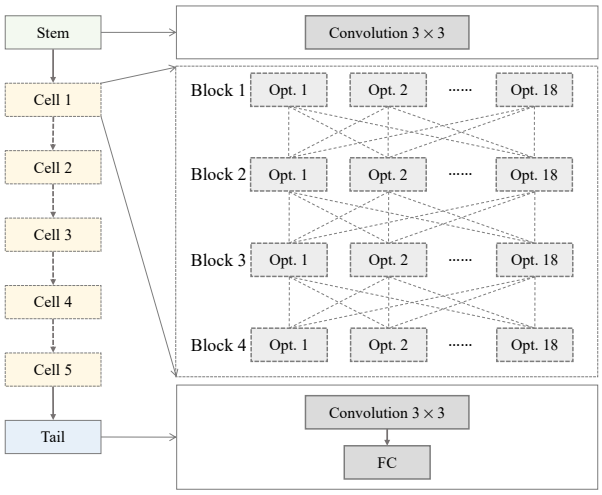

Figure 2: The individual's architecture of EOFGA.

3.4 Individual Evaluation

In evolutionary optimization, candidate individuals (a.k.a. subnets) are evaluated to obtain their corresponding fitness values. In contrast to random weight initialization, EOFGA utilizes the weights inherited from the supernet. Subsequently, the subnet is further trained on the training data for one epoch, thus enabling all training data to traverse and optimize the weights. The fitness value of the subnet is then assessed on the fitness evaluation data, with the classification accuracy used as the fitness metric.

Please note that most publicly available datasets (such as CIFARs and ImageNet) are formally divided into a training set $\mathcal{D}_{TRA}$ and a test set $\mathcal{D}_{TEST}$, and we further divide the public training set $\mathcal{D}_{TRA}$ into training data $\mathcal{D}_{train}$ and fitness evaluation data $\mathcal{D}_{eva}$ according to the ratio of 8:2, and there is no intersection between them, i.e.,

$$\mathcal{D}_{train} \cup \mathcal{D}_{eva} = \mathcal{D}_{TRA}, \quad \mathcal{D}_{train} \cap \mathcal{D}_{eva} = \emptyset \, . \tag{4}$$

The training data $\mathcal{D}_{train}$ is used for training the weights, and the fitness evaluation data $\mathcal{D}_{eva}$ is used for the assessment of the candidate's performance.

3.5 Parent Selection

After the fitness evaluation of the current population is finished, the parents need to be selected according to the parent selection strategy: EOFGA employs a kind of elitism strategy, i.e., half of the individuals with the highest fitness are chosen as the parents to form the mating pool, and the remaining individuals are eliminated.

In many cases, a tournament selection strategy is preferred to select the parent individuals to ensure the diversity of the mating pool. If only the individuals of highest fitness are chosen, some potentially advantageous genes may be ruled out, and the GA may get into premature convergence [1]. However, as mentioned before, the evaluated fitness may not always accurately reflect the candidate's true performance in one-shot NAS. Although EOFGA attempts to make the evaluation more reliable, the evaluation bias

still exists. Thus, even though only the top half of individuals are selected according to the fitness values, not all of the corresponding true fitnesses are in the top half, which is expected to improve the diversity and help avoid premature convergence.

3.6 Offspring Generation

In EOFGA, the proposed crossover and mutation strategies are operated on the parent individuals to generate offspring. The developed crossover and mutation methods are detailed as follows.

3.6.1 *Cell-based Crossover.*

As the main body of networks is built by cells in EOFGA, the basic unit for crossover between two parents is the cell. Specifically, the total number of crossed cells is randomly selected between one and two, and the index(es) of the crossed cell(s) is/are selected randomly. Then, the chosen cell(s) of the same index in the two parents swap with each other, and two new subnets are generated. Finally, one subnet is randomly selected and preserved. Figure 3 shows an example of the cell-based crossover, where the indexes of two and four are selected, and the corresponding cells are exchanged between the two parent individuals, resulting in two new subnets that contain cells from both the two parents.

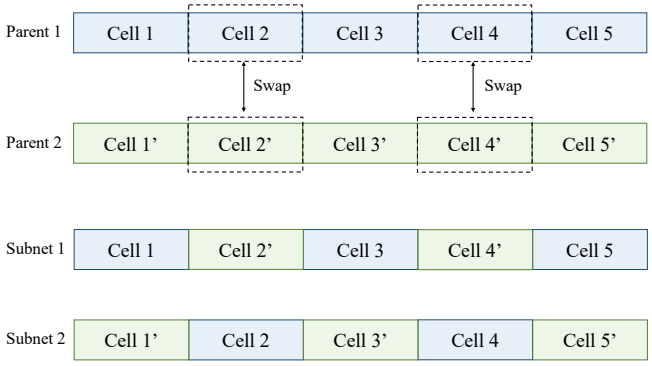

Figure 3: An example of the cell-based crossover.

3.6.2 *Block-based Mutation.*

A block-based mutation method is developed to further modify the individual, improving the exploration of the search. In one cell, there are at most four blocks, and the mutation operation aims to facilitate the exploration of the block structures. Algorithm 1 provides the details of the block-based mutation method. The number of mutation operations n_m is determined first and is between 1 and 3 (line 1). For each mutation operation, a specific cell is selected (line 3), a block inside the cell is chosen (line 4), and the number of blocks n_b of the cell is specified (line 5). Subsequently, a type of mutation is randomly selected among three operations: *adding*, *removing*, and *modifying* (line 6). If *adding* is selected and the number of blocks is lower than the maximal number of blocks 4, a new block will be generated and added next to the chosen block (lines 7-9); if *removing* is chosen and there are more than one block in the cell, the selected block will be removed (lines 10-11); otherwise, a new randomly generated block will replace the chosen block (lines 12-14).

Algorithm 1: The Block-based Mutation

Input: The subnet s.
Output: The mutated subnet s.

1 $n_m \leftarrow$ determine the number of mutation operations;
2 **for** $i = 1;\ i \leq n_m;\ i \leftarrow i + 1$ **do**
3 $cell \leftarrow$ Randomly select a cell from s;
4 $block \leftarrow$ Randomly select a block from $cell$;
5 $n_b \leftarrow$ the number of blocks in $cell$;
6 $operation \leftarrow$ Randomly select one from {*adding*, *removing*, *modifying*};
7 **if** $operation$ is *adding* and $n_b < 4$ **then**
8 $block_{new} \leftarrow$ generate a new block;
9 Add $block_{new}$ next to $block$;
10 **else if** $operation$ is *removing* and $n_b > 1$ **then**
11 Remove $block$ from $cell$;
12 **else**
13 $block_{new} \leftarrow$ generate a new block;
14 Replace $block$ with $block_{new}$;
15 **end**
16 **end**

3.7 Supernet Fine-Tuning

The weights of the supernet are fine-tuned for the focused search region of each generation, providing more reliable fitness evaluations, since the individuals for evaluation are distributed within the region. During the fine-tuning process, the supernet is trained on a selected path within the region for each batch of data, continuing until the predefined number of training epochs or the stopping criterion is met.

A novel path sampling strategy is proposed in order to determine the path, which reflects the distribution of the offspring in the focused search region. Specifically, this strategy utilizes an offspring generation method to sample the path: an individual is generated for each batch of data, and its corresponding path is activated for fine-tuning, thus allowing the weights of the supernet in the focused search region to be fine-tuned.

3.8 New Population

Figure 4 illustrates the process of new population generation in EOFGA. Specifically, the top half of individuals of the current population are selected as parents. A small part of the new population is built by choosing some of the top parent individuals, while the remaining part is comprised of offspring produced by crossover and mutation from the parents. This is different from standard GAs, which employ an environmental selection from both the current population and the offspring population to form the population of the next generation.

The reason for this difference is that the parents and offspring are evaluated under different evaluation conditions, i.e., the supernet is further fine-tuned for the offspring evaluation, and thus it would be unfair to compare the fitness of parents and offspring together. Therefore, if following the standard way to form the new population, both the parents and offspring need to be evaluated after the supernet is fine-tuned. This would require all the parents

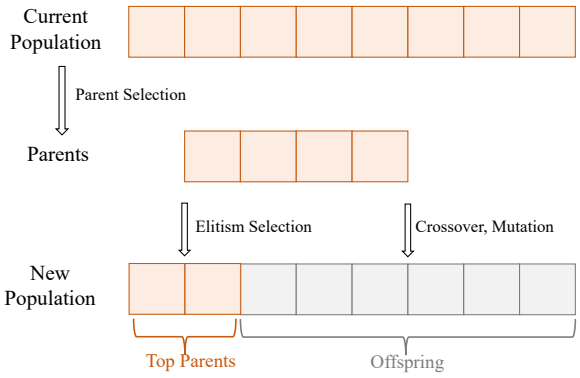

Figure 4: New population generation method in EOFGA.

to be re-evaluated, resulting in increased computational resources. To avoid this, EOFGA directly selects some parents with the highest fitness to build a part of the new population, with the number of selected parents being 10% of the population size.

4 EXPERIMENT DESIGN

4.1 Benchmark Datasets

In order to assess the performance of EOFGA, three benchmark datasets are employed: CIFAR10, CIFAR100, and ImageNet. Their selection is based on two criteria: (1) they present different levels of difficulty in terms of the number of images and classification categories, providing a comprehensive evaluation of EOFGA's performance; (2) they are popular public datasets, allowing comparison of EOFGA with other peer competitors.

Specifically, both CIFAR10 and CIFAR100 contain 60,000 RGB images of size 32×32. Each dataset has a training set of 50,000 images and a test set of 10,000 images. CIFAR10 comprises 10 classes, each with an equal number of images, while CIFAR100 includes 100 classes. Conversely, ImageNet is a more challenging dataset with over 14,000,000 images categorized into 1,000 classes, of which 13,000,000 are used for training. The images in ImageNet are much larger than those of CIFAR10 and CIFAR100, with a resolution of 224×224.

4.2 Parameter Settings

EOFGA includes the supernet training and the search process, and the search comprises the supernet fine-tuning and the developed GA procedure. For the convenience of comparing with peer competitors, the final searched architecture needs further evaluation, a.k.a. the post-search evaluation. The parameter settings for the above processes are outlined separately.

For the supernet training, the total number of training epochs is 100 for both CIFAR10 and CIFAR100 and is 50 for ImageNet. Following [9], a cosine annealing learning rate policy is employed with an initial learning rate of 0.025 and a batch size of 96, which is a relatively small number. This is because different sampling paths are selected for different data batches, and a small batch size can sample more paths.

While fine-tuning the supernet, most settings are similar to the supernet training process, except for the stopping criterion. The maximal number of fine-tune epochs is 30 for CIFARs and 10 for ImageNet, and it will terminate if the training loss does not decrease for three consecutive epochs, implying the weights are converged.

For the developed GA process, the population size is 50, which is a relatively big number in the field of NAS since the selected parent individuals help build the refined regions, and a larger number can help build a delicate region. The number of generations is 10 considering the computational cost. In principle, a larger number of generations can lead to better performance. This setting is a trade-off between efficiency and performance.

The post-search evaluation involves training the searched architecture for 300 epochs using SGD search. A cosine annealing learning rate policy is implemented with an initial learning rate of 0.01 and a batch size of 96. For CIFAR10 and CIFAR100, a dropout rate of 0.2 is applied to the fully-connected layers. However, dropout is not used on ImageNet, as the dataset is complex and the chances of overfitting are small. Besides, an SE layer [11] is added to each block of the searched architecture to improve the accuracy on ImageNet.

5 RESULTS AND ANALYSIS

5.1 Overall Performance

EOFGA is performed on CIFAR10, CIFAR100, and ImageNet datasets. For CIFAR10 and CIFAR100, we compared the classification error rate, the model size (measured by the number of parameters), and the searching cost (measured by GPU-days) with peer competitors. On ImageNet, besides the above three measurements, we also compare the models' computation complexity (measured by the number of FLOPs). We performed the experiment five times on CIFAR10 and CIFAR100, but due to the limited computing budget, only one run was conducted on ImageNet.

5.1.1 Performance on CIFAR10.

Table 1 presents the experimental results on CIFAR10. Specifically, *EOFGA (best)* has the lowest classification error rate, *EOFGA (median)* has the median error rate, and *EOFGA (mean)* represents the average error rate. GPU-days include the supernet's initial training and the developed GA search process. Comparisons are made with 26 peer competitors, including three manually-designed networks and 23 NAS methods.

In the best case, EOFGA achieves an error rate of 2.50%, and the model size is 2.16M. The computational cost is 0.54 GPU-days. Its error rate is only 0.01% lower than PRE-NAS's [20], which ranks second. Proxyless NAS [5] has the lowest error rate, but with both the number of parameters and the searching cost much higher than EOFGA's. EOFGA (median) achieves an error rate of 2.56%, along with a model size of 2.13M and a computational cost of 0.50 GPU-days. The performance of EOFGA (mean) is similar to EOFGA (median) for the three measurements. We mainly use EOFGA (median) for comparisons with other competitors since it can represent a more general result of EOFGA.

Considering the classification accuracy, EOFGA (median) ranks third out of the 26 competitors. PRE-NAS's [20] error rate of 2.49% is

Table 1: The comparisons on the CIFAR10 dataset.

Model	Error Rate	#Parameters	GPU-Days
FractalNet [13]	5.22%	38.6M	—
ResNet-101 [10]	6.43%	1.7M	—
DenseNet (k=24) [12]	3.74%	27.2M	—
CGP-CNN [30]	5.98%	2.64M	27
NAS [46]	6.01%	2.5M	22,400
Large-scale Evolution [23]	5.4%	5.4M	2,750
Block-QNN-S [44]	4.38%	6.1M	90
PNASNet-5 [15]	3.41%	3.2M	150
AmoebaNet-B [22]	2.98%	34.9M	3,150
CARS [37]	2.62%	3.6M	**0.4**
PRE-NAS [20]	2.49%	4.5M	0.6
NSGA-Net [18]	2.75%	3.3 M	4
LEMONADE [8]	2.58%	13.1M	90
NSGANetV1-A1 [19]	3.49%	**0.5M**	27
Proxyless NAS [5]	**2.08%**	5.7M	1,500
EAS [4]	4.23%	23.4M	<10
NASNET-A [47]	2.97%	27.6M	2,000
AECNN [32]	4.3%	2.0M	27
EffPnet [34]	3.58%	2.68M	<3
DARTS [16]	2.82%	3.4M	1
EIGEN [24]	5.4%	2.6M	2
CNN-GA [33]	4.78%	2.9M	35
SNAS [35]	2.85%	2.8M	1.5
ENAS [21]	2.89%	4.6M	0.5
RandomNAS [41]	2.59%	3.1M	0.7
Modulenet [6]	2.77%	—	2.0
EOFGA (best)	2.50%	2.16M	0.54
EOFGA (median)	2.56%	2.13M	0.50
EOFGA (mean)	2.59%	2.11M	0.49

only 0.07% lower than EOFGA's, but the model size of 4.5M is more than twice that of EOFGA's 2.13M; the computational cost is also slightly higher than EOFGA's. The error rate of Proxyless NAS [5] is 0.46% lower than EOFGA, but its model size is 2.7 times larger than EOFGA's and its computational cost is 3000 times higher. As for the model size, EOFGA's 2.13M is the fourth smallest. ResNet-101 [10], NSGANetV1-A1 [19], and AECNN [32] are all of smaller model sizes, but their classification performance is inferior to EOFGA's. As for the computational cost, EOFGA only requires 0.5 GPU-days, which is slightly larger than CARS's [37] 0.4 GPU-day. However, CARS's error rate is higher and the model size is larger. Overall, EOFGA achieves promising results on CIFAR-10: the classification rate is very high within only half a GPU-day; meanwhile, the model size is smaller than most of its peer competitors, implying that EOFGA achieves an outstanding balance between effectiveness and efficiency.

5.1.2 *Performance on CIFAR100.*

Table 2 presents the performance of EOFGA on CIFAR100. Specifically, EOFGA (best) achieves an error rate of 16.85% with a model size of 2.26M and a computational cost of 0.83 GPU-days; EOFGA (median) has an error rate of 17.07%, a model size of 1.96M, and a computational cost of 0.92 GPU-days; EOFGA (mean) achieves an error rate of 17.23% within 0.94 GPU-days. Similarly, we mainly discuss the median EOFGA with other competitors.

Table 2: The comparisons on the CIFAR100 dataset.

Model	Error Rate	#Parameters	GPU-Days
FractalNet [13]	22.3%	38.6M	—
ResNet-101 [10]	25.16%	1.7M	—
DenseNet (k=40) [12]	17.2%	25.6M	—
Large-scale Evolution [23]	23%	40.4M	2,730
Block-QNN-S [44]	20.65%	6.1M	90
MetaQNN [2]	27.14%	—	100
AmoebaNet-A [22]	18.93%	3.1M	3,150
DARTS [16]	17.54%	3.4M	1
NSGANetV1-A1 [19]	19.23%	**0.7M**	27
AE-CNN+E2EPP [31]	22.02%	20.9M	10
AECNN [32]	22.40%	5.4M	36
EIGEN [24]	21.9%	11.8M	5
CNN-GA [33]	20.03%	4.1M	40
NSGA-Net [18]	20.74%	3.3M	8
PNASNet-5 [15]	19.53%	3.2M	150
ENAS [21]	19.43%	4.6M	**0.5**
SI-EvoNet-S [40]	**15.70%**	3.32M	0.81
Modulenet [6]	17.76%	—	—
EOFGA (best)	16.85%	2.26M	0.83
EOFGA (median)	17.07%	1.96M	0.92
EOFGA (mean)	17.23%	2.18M	0.94

EOFGA (median)'s error rate ranks second over 18 peer competitors, and only SI-EvoNet-S's [40] error rate is 1.37% lower than it, but the model size is 1.7 times EOFGA's 1.96M. As for the model size, EOFGA has the third smallest number of parameters; ResNet-101 [10] and NSGANetV1-A1 [19] have smaller model sizes, but their error rates are higher than EOFGA. In terms of computational cost, EOFGA's 0.92 GPU-days ranks third; ENAS's 0.5 GPU-days and SI-EvoNet-S's [40] 0.81 GPU-days are lower, but ENAS has a worse classification performance and SI-EvoNet-S has a larger model size. Therefore, EOFGA obtains a high classification accuracy within a very low memory consumption and computational cost, demonstrating its outstanding performance on CIFAR100.

5.1.3 *Performance on ImageNet.*

Table 3 presents the comparison results of EOFGA with other competitors in terms of the Top-1 and Top-5 error rates, model scale, model complexity, and searching cost. We can see that the Top-1 error rate of EOFGA is 0.6% higher than NSGANetV1-A3 and 0.1% higher than AmoebaNet-C. On the other hand, the Top-5 error rate of EOFGA is 0.54% higher than NSGANetV1-A3 and 0.1% lower than AmoebaNet-C. NSGANetV1-A3 has a larger computing complexity and 3.38 times higher searching cost than EOFGA, while AmoebaNet-C has a larger model size, higher complexity, and significantly higher searching cost. In terms of computing complexity, EOFGA's number of FLOPs of 455M ranks second, only behind MobileNetV3, with a Top-1 error rate 0.4% lower than EOFGA. Additionally, EOFGA achieved satisfying results on ImageNet with a small computational cost of 8.0 GPU-days. Although DARTS and SNAS have less computational cost, they have much worse classification performance than EOFGA. Thus, EOFGA demonstrated promising results on ImageNet, obtaining satisfactory classification

performance with a less complex model and reasonable computational cost.

Table 3: The comparisons on the ImageNet dataset.

Model	Top-1 Err.	Top-5 Err.	#Params	#FLOPs	GPU-Days
ResNet-34 [10]	26.8%	8.7%	21.8M	7,360M	—
MobileNetV2 [27]	28.0%	9.0%	**3.4M**	600M	—
ShuffleNet [42]	26.3%	—	5.4M	524M	—
MobileNetV3 [11]	24.8%	—	5.4M	**450M**	—
Proxyless NAS [5]	24.9%	7.5%	7.1M	—	8.3
PNASNet-5 [15]	25.8%	8.1%	5.1M	588M	150
AmoebaNet-C [22]	24.3%	7.6%	6.4M	570M	3,150
NSGANetV1-A3 [19]	**23.8%**	**7.0%**	5.0M	570M	27
DARTS [16]	26.7%	8.6%	4.7M	—	4
SNAS [35]	27.3%	9.2%	4.3M	—	**1.5**
EOFGA	24.4%	7.4%	5.7M	455M	8.0

5.2 Effectiveness of Supernet Fine-Tuning

To prove the effectiveness of the proposed supernet fine-tuning method, we design a comparative experiment by developing an algorithm without supernet fine-tuning, termed EOWGA. EOWGA has the same setting as EOFGA except for removing the fine-tuning of the supernet. EOWGA is performed on both CIFAR10 and CIFAR100 to test the performance and compare it with EOFGA.

Table 4 shows the comparison results of EOFGA and EOWGA. On CIFAR10, EOWGA's error rate is 2.90%, which is 0.34% higher than EOFGA. The model sizes are similar, and EOWGA's computational cost is 0.1 GPU-days less than EOFGA's 0.5 GPU-days. On CIFAR100, EOWGA's classification accuracy is 1.63% worse than EOFGA, along with a larger model scale, but the computational cost is 0.25 GPU-days less. Therefore, the searched architectures of EOWGA are worse than EOFGA's in terms of both error rate and model size. The results of this comparison demonstrate that the proposed supernet fine-tuning method is effective in helping to search for better architectures, despite consuming more computational resources. Nevertheless, the total consumed time is still low compared to many existing methods, as shown in Tables 1 and 2.

Table 4: The comparisons of EOFGA and EOWGA.

Method	Error Rate	Dataset	#Params	GPU-Days
EOFGA	2.56%	CIFAR10	2.13M	0.50
	17.07%	CIFAR100	1.96M	0.92
EOWGA	2.90%	CIFAR10	2.14M	0.40
	18.69%	CIFAR100	2.41M	0.67

5.3 Analysis of Supernet Fine-Tuning

During the evolution, the supernet is fine-tuning for each generation in EOFGA, and the fine-tuning will stop automatically if the weights converge to a certain degree (training loss no longer decreases). The sampling regions of the supernet of each generation are also getting smaller and more promising along with the evolution, and it is worth analyzing the fine-tuning epochs of each generation.

Figure 5 shows the supernet fine-tuning epochs for each generation of EOFGA on CIFAR10. It can be observed that the number of epochs is 30 (the maximum) in the first four generations and gets smaller in the next four generations. From the eighth generation, the number of epochs tends to be 3. The tendency implies that the weights of the supernet get close to the optimum along with the evolutionary process. In this way, the supernet can provide more reliable fitness estimations for the individuals, supporting the proposed supernet fine-tuning strategy.

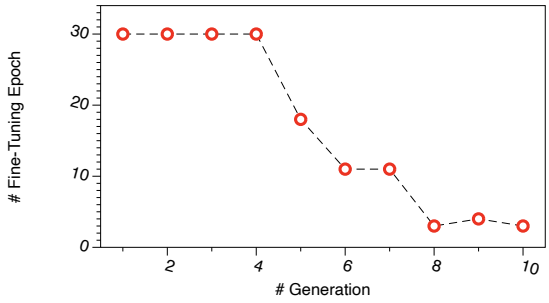

Figure 5: The supernet fine-tuning epochs for each generation on CIFAR10.

6 CONCLUSIONS

This paper aims to propose an effective one-shot NAS method for image classification. The goal has been achieved by proposing a supernet fine-tuning strategy along with the evolution and developing a GA-based search paradigm to support the supernet fine-tuning. Specifically, a path-sampling strategy is designed for the supernet fine-tuning and offspring production. A new population generation method is proposed to avoid unfair fitness comparisons between parents and offspring. The proposed EOFGA method is examined on three popular benchmark datasets: CIFAR10, CIFAR100, and ImageNet, and is compared with 32 peer competitors in terms of the searched models' performance (measured by the classification accuracy, model size, and computation complexity) and the algorithms' computational cost. EOFGA achieves promising performance based on the above measurements; in particular, the computational cost is just 0.50 GPU-days on CIFAR10 and 0.92 GPU-days on CIFAR100, significantly lower than most competitors. Further experiments verify the effectiveness of the supernet fine-tuning and show the weights of the supernet are getting convergence by the fine-tuning.

For EOFNAS, the searched network is of high classification accuracy but is still a "black box", and it is hard to understand why such an architecture can bring high accuracy. In future work, we plan to investigate the interpretability of the network architecture and reveal the relationship between the performance and the architecture of the searched network.

REFERENCES

[1] Chang Wook Ahn and R.S. Ramakrishna. 2003. Elitism-based compact genetic algorithms. *IEEE Transactions on Evolutionary Computation* 7, 4 (2003), 367–385. https://doi.org/10.1109/TEVC.2003.814633

[2] Bowen Baker, Otkrist Gupta, Nikhil Naik, and Ramesh Raskar. 2016. Designing neural network architectures using reinforcement learning. *arXiv preprint arXiv:1611.02167* (2016).

[3] Gabriel Bender, Pieter-Jan Kindermans, Barret Zoph, Vijay Vasudevan, and Quoc Le. 2018. Understanding and Simplifying One-Shot Architecture Search. In *Proceedings of the 35th International Conference on Machine Learning (Proceedings of Machine Learning Research, Vol. 80)*, Jennifer Dy and Andreas Krause (Eds.). PMLR, 550–559. https://proceedings.mlr.press/v80/bender18a.html

[4] Han Cai, Tianyao Chen, Weinan Zhang, Yong Yu, and Jun Wang. 2018. Efficient architecture search by network transformation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 32.

[5] Han Cai, Ligeng Zhu, and Song Han. 2018. Proxylessnas: Direct neural architecture search on target task and hardware. *arXiv preprint arXiv:1812.00332* (2018).

[6] Yaran Chen, Ruiyuan Gao, Fenggang Liu, and Dongbin Zhao. 2021. ModuleNet: Knowledge-Inherited Neural Architecture Search. *IEEE Transactions on Cybernetics* (2021), 1–11. https://doi.org/10.1109/TCYB.2021.3078573

[7] Xiangxiang Chu, Bo Zhang, and Ruijun Xu. 2021. Fairnas: Rethinking evaluation fairness of weight sharing neural architecture search. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 12239–12248.

[8] Thomas Elsken, Jan Hendrik Metzen, and Frank Hutter. 2018. Efficient multi-objective neural architecture search via lamarckian evolution. *arXiv preprint arXiv:1804.09081* (2018).

[9] Zichao Guo, Xiangyu Zhang, Haoyuan Mu, Wen Heng, Zechun Liu, Yichen Wei, and Jian Sun. 2020. Single path one-shot neural architecture search with uniform sampling. In *European conference on computer vision*. Springer, 544–560.

[10] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 770–778.

[11] Andrew G. Howard, Mark Sandler, Grace Chu, Liang-Chieh Chen, Bo Chen, Mingxing Tan, Weijun Wang, Yukun Zhu, Ruoming Pang, Vijay Vasudevan, Quoc V. Le, and Hartwig Adam. 2019. Searching for MobileNetV3. *2019 IEEE/CVF International Conference on Computer Vision (ICCV)* (2019), 1314–1324.

[12] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. 2017. Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 4700–4708.

[13] Gustav Larsson, Michael Maire, and Gregory Shakhnarovich. 2016. Fractalnet: Ultra-deep neural networks without residuals. *arXiv preprint arXiv:1605.07648* (2016).

[14] Yann LeCun, Bernhard Boser, John S Denker, Donnie Henderson, Richard E Howard, Wayne Hubbard, and Lawrence D Jackel. 1989. Backpropagation applied to handwritten zip code recognition. *Neural computation* 1, 4 (1989), 541–551.

[15] Chenxi Liu, Barret Zoph, Maxim Neumann, Jonathon Shlens, Wei Hua, Li-Jia Li, Li Fei-Fei, Alan Yuille, Jonathan Huang, and Kevin Murphy. 2018. Progressive neural architecture search. In *Proceedings of the European Conference on Computer Vision (ECCV)*. 19–34.

[16] Hanxiao Liu, Karen Simonyan, and Yiming Yang. 2018. Darts: Differentiable architecture search. *arXiv preprint arXiv:1806.09055* (2018).

[17] Shiqing Liu, Haoyu Zhang, and Yaochu Jin. 2022. A survey on computationally efficient neural architecture search. *Journal of Automation and Intelligence* 1, 1 (2022), 100002. https://doi.org/10.1016/j.jai.2022.100002

[18] Zhichao Lu, Ian Whalen, Vishnu Boddeti, Yashesh Dhebar, Kalyanmoy Deb, Erik Goodman, and Wolfgang Banzhaf. 2019. Nsga-net: neural architecture search using multi-objective genetic algorithm. In *Proceedings of the Genetic and Evolutionary Computation Conference*. 419–427.

[19] Zhichao Lu, Ian Whalen, Yashesh Dhebar, Kalyanmoy Deb, Erik D. Goodman, Wolfgang Banzhaf, and Vishnu Naresh Boddeti. 2021. Multi-Objective Evolutionary Design of Deep Convolutional Neural Networks for Image Classification. *IEEE Transactions on Evolutionary Computation* 25, 2 (2021), 277–291. https://doi.org/10.1109/TEVC.2020.3024708

[20] Yameng Peng, Andy Song, Vic Ciesielski, Haytham M Fayek, and Xiaojun Chang. 2022. PRE-NAS: Predictor-assisted Evolutionary Neural Architecture Search. *arXiv preprint arXiv:2204.12726* (2022).

[21] Hieu Pham, Melody Guan, Barret Zoph, Quoc Le, and Jeff Dean. 2018. Efficient neural architecture search via parameters sharing. In *International Conference on Machine Learning*. PMLR, 4095–4104.

[22] Esteban Real, Alok Aggarwal, Yanping Huang, and Quoc V Le. 2019. Regularized evolution for image classifier architecture search. In *Proceedings of the aaai conference on artificial intelligence*, Vol. 33. 4780–4789.

[23] Esteban Real, Sherry Moore, Andrew Selle, Saurabh Saxena, Yutaka Leon Suematsu, Jie Tan, Quoc V Le, and Alexey Kurakin. 2017. Large-scale evolution of image classifiers. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*. 2902–2911.

[24] Jian Ren, Zhe Li, Jianchao Yang, Ning Xu, Tianbao Yang, and David J Foran. 2019. Eigen: Ecologically-inspired genetic approach for neural network structure searching from scratch. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 9059–9068.

[25] Pengzhen Ren, Yun Xiao, Xiaojun Chang, Po-yao Huang, Zhihui Li, Xiaojiang Chen, and Xin Wang. 2021. A Comprehensive Survey of Neural Architecture Search: Challenges and Solutions. *ACM Comput. Surv.* 54, 4, Article 76 (may 2021), 34 pages. https://doi.org/10.1145/3447582

[26] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. 1986. Learning representations by back-propagating errors. *Nature* 323, 6088 (1986), 533–536.

[27] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. 2018. Mobilenetv2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 4510–4520.

[28] Yao Shu, Wei Wang, and Shaofeng Cai. 2019. Understanding architectures learnt by cell-based neural architecture search. *arXiv preprint arXiv:1909.09569* (2019).

[29] Xiu Su, Shan You, Mingkai Zheng, Fei Wang, Chen Qian, Changshui Zhang, and Chang Xu. 2021. K-shot NAS: Learnable Weight-Sharing for NAS with K-shot Supernets. In *Proceedings of the 38th International Conference on Machine Learning (Proceedings of Machine Learning Research, Vol. 139)*, Marina Meila and Tong Zhang (Eds.). PMLR, 9880–9890. https://proceedings.mlr.press/v139/su21a.html

[30] Masanori Suganuma, Shinichi Shirakawa, and Tomoharu Nagao. 2017. A genetic programming approach to designing convolutional neural network architectures. In *Proceedings of the genetic and evolutionary computation conference*. 497–504.

[31] Yanan Sun, Handing Wang, Bing Xue, Yaochu Jin, Gary G Yen, and Mengjie Zhang. 2019. Surrogate-assisted evolutionary deep learning using an end-to-end random forest-based performance predictor. *IEEE Transactions on Evolutionary Computation* 24, 2 (2019), 350–364.

[32] Yanan Sun, Bing Xue, Mengjie Zhang, and Gary G Yen. 2019. Completely automated CNN architecture design based on blocks. *IEEE transactions on neural networks and learning systems* 31, 4 (2019), 1242–1254.

[33] Yanan Sun, Bing Xue, Mengjie Zhang, Gary G. Yen, and Jiancheng Lv. 2020. Automatically Designing CNN Architectures Using the Genetic Algorithm for Image Classification. *IEEE Transactions on Cybernetics* 50, 9 (2020), 3840–3854. https://doi.org/10.1109/TCYB.2020.2983860

[34] Bin Wang, Bing Xue, and Mengjie Zhang. 2022. Surrogate-Assisted Particle Swarm Optimization for Evolving Variable-Length Transferable Blocks for Image Classification. *IEEE Transactions on Neural Networks and Learning Systems* 33, 8 (2022), 3727–3740. https://doi.org/10.1109/TNNLS.2021.3054400

[35] Sirui Xie, Hehui Zheng, Chunxiao Liu, and Liang Lin. 2018. SNAS: stochastic neural architecture search. *arXiv preprint arXiv:1812.09926* (2018).

[36] Antoine Yang, Pedro M Esperança, and Fabio M Carlucci. 2019. NAS evaluation is frustratingly hard. *arXiv preprint arXiv:1912.12522* (2019).

[37] Zhaohui Yang, Yunhe Wang, Xinghao Chen, Boxin Shi, Chao Xu, Chunjing Xu, Qi Tian, and Chang Xu. 2020. CARS: Continuous Evolution for Efficient Neural Architecture Search. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

[38] Shan You, Tao Huang, Mingmin Yang, Fei Wang, Chen Qian, and Changshui Zhang. 2020. GreedyNAS: Towards Fast One-Shot NAS With Greedy Supernet. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

[39] Kaicheng Yu, Christian Suito, Martin Jaggi, Claudiu-Cristian Musat, and Mathieu Salzmann. 2020. Evaluating the search phase of neural architecture search. In *ICRL 2020 Eighth International Conference on Learning Representations*.

[40] Haoyu Zhang, Yaochu Jin, Ran Cheng, and Kuangrong Hao. 2020. Efficient evolutionary search of attention convolutional networks via sampled training and node inheritance. *IEEE Transactions on Evolutionary Computation* 25, 2 (2020), 371–385.

[41] Miao Zhang, Huiqi Li, Shirui Pan, Xiaojun Chang, Chuan Zhou, Zongyuan Ge, and Steven Su. 2020. One-shot neural architecture search: Maximising diversity to overcome catastrophic forgetting. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 43, 9 (2020), 2921–2935.

[42] Xiangyu Zhang, Xinyu Zhou, Mengxiao Lin, and Jian Sun. 2018. Shufflenet: An extremely efficient convolutional neural network for mobile devices. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 6848–6856.

[43] Yiyang Zhao, Linnan Wang, Yuandong Tian, Rodrigo Fonseca, and Tian Guo. 2021. Few-Shot Neural Architecture Search. In *Proceedings of the 38th International Conference on Machine Learning (Proceedings of Machine Learning Research, Vol. 139)*, Marina Meila and Tong Zhang (Eds.). PMLR, 12707–12718. https://proceedings.mlr.press/v139/zhao21d.html

[44] Zhao Zhong, Junjie Yan, Wei Wu, Jing Shao, and Cheng-Lin Liu. 2018. Practical block-wise neural network architecture generation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2423–2432.

[45] Zixuan Zhou, Xuefei Ning, Yi Cai, Jiashu Han, Yiping Deng, Yuhan Dong, Huazhong Yang, and Yu Wang. 2022. CLOSE: Curriculum Learning on the Sharing Extent Towards Better One-Shot NAS. In *Computer Vision – ECCV 2022*, Shai Avidan, Gabriel Brostow, Moustapha Cissé, Giovanni Maria Farinella, and Tal Hassner (Eds.). Springer Nature Switzerland, Cham, 578–594.

[46] Barret Zoph and Quoc V Le. 2016. Neural architecture search with reinforcement learning. *arXiv preprint arXiv:1611.01578* (2016).

[47] Barret Zoph, Vijay Vasudevan, Jonathon Shlens, and Quoc V Le. 2018. Learning transferable architectures for scalable image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 8697–8710.