

LoNAS: Low-Cost Neural Architecture Search Using a Three-Stage Evolutionary Algorithm

Abstract

Neural architecture search (NAS) has been widely studied to design high-performance network architectures automatically. However, existing approaches require more search time and substantial resource consumption due to their intensive architecture evaluations. Moreover, recently developed NAS algorithms are noncompetitive when combining multiple competing and conflicting objectives, e.g., the test accuracy and the number of parameters. In this paper, a low-cost NAS (LoNAS) method is proposed to address these problems. First, a variable-architecture encoding strategy based on a novel Reg Block is designed to construct high accuracy network architectures with few parameters. Second, a training-free proxy based on the neural tangent kernel (NTK) is proposed to accelerate the search process efficiently. Finally, a three-stage evolutionary algorithm (EA) based on multiple-criteria environmental selection and a set of block-based mutation operators are designed to balance exploration and exploitation better. The experimental results show that LoNAS finds network architectures with competitive performance compared to the state-of-the-art architectures in test accuracy and the number of parameters. Moreover, LoNAS uses less search time and fewer computational resources, consuming only 0.02 GPU Days with one



©SHUTTERSTOCK.COM/W001X

GPU on CIFAR-10 and CIFAR-100. Furthermore, the architectures found by LoNAS on CIFAR-10 and CIFAR-100 exhibit good transferability to ImageNet-16-120, with the test accuracy surpassing that of the state-of-the-art network architectures.

1. Introduction

Deep learning has made substantial progress in various computer vision tasks. Manually-designed network architectures, such as VGGNet [1], ResNet [2], Inception [3], and DenseNet [4], are some of the essential driving forces in developing this research field. While manually-designed network architectures can achieve outstanding classification performance, the design process requires professional domain knowledge, which is possessed by only a few experts. Moreover, manually designed methods consume considerable time and computational resources due to the need for repeated experiments.

Numerous studies on *neural architecture search* (NAS) have recently been performed to design neural networks automatically. NAS algorithms allow tuning network architecture skills to be transparent for users who need to be more familiar with domain knowledge. The required human efforts and costs can be reduced by automating NAS algorithms. NAS can be divided into two categories, based on whether extra manual fine-tuning is required or not. The first category is semiautomatic NAS algorithms, which needs to be combined with manual fine-tuning. For example, EAS [5] is based on a good network to implement search. The base network needs to be designed manually based on expertise. Smash [6] needs to manually train a SuperNet, and then the search process can be executed in the SuperNet. The second category is automatic NAS algorithms, which can be used without manual fine-tuning, such as [7], [8], [9]. The network architectures found by NAS algorithms outperform those by manually-designed algorithms and have

been applied in many fields [10], [11], [12]. However, the search time and computational resource costs of NAS algorithms are usually high. Most existing NAS algorithms rely on validation datasets to optimize network architectures, requiring considerable time and intensive computational resources; for example, NASNet [13] uses 500 GPUs across four days.

The architecture search problem is usually framed as a single-objective optimization problem that cannot simultaneously consider multiple objectives [9], [13], [14]. Most real-world deployments need high classification performance and low computational resource usage (e.g., model size and computational complexity). Several manually-designed network architectures, such as MobileNet [15] and MobileNetV2 [16], have been designed to reduce computational costs while attaining high classification performance. Some recent NAS algorithms based on multi-objective optimization have been proposed to develop network architectures that are easy to calculate and deploy. In [17], NSGA-Net considered the trade-off between test accuracy and computational complexity. In [18], the classification performance and the number of parameters were combined as objectives. However, [17], [18] still require numerous computational resources and long search time. In this paper, a low-cost NAS (LoNAS) method based on a three-stage evolutionary algorithm (EA) [19] is proposed. This approach significantly reduces search time and computational resource consumption. Moreover, the classification performance and the number of parameters can be effectively balanced in LoNAS. The main contributions of the proposed algorithm are summarized as follows.

- 1) A novel network block called the Reg Block is proposed in LoNAS. The Reg Block includes the group convolution and SENet module, reducing the number of parameters and improving the network's classification performance. A variable-architecture encoding strategy is designed based on the Reg Block to encode the network architecture. By simultaneously considering variable groups, group widths,

SENet modules, network lengths, and pooling layer strides, an expanded search space can be constructed. Then, more network architectures with good performance can be found in the expanded search space.

- 2) A training-free proxy based on the neural tangent kernel (NTK) is designed to evaluate the performance of individuals in the population. The NTK can effectively characterize the trainability of a network architecture since the condition number of the NTK (K_N) is negatively correlated with the test accuracy. As K_N can be computed without training, the search time and computational resources can be significantly reduced.
- 3) A three-stage EA based on a multiple-criteria environmental selection strategy is proposed to effectively balance exploration and exploitation. The environmental selection process criteria are based on the K_N and individual lifespans. The lifespan is a property associated with each individual that indicates the number of evolution generations an individual goes through. In the first and third stages, K_N is used as the criterion for eliminating individuals. In the second stage, individuals are selected according to their lifespans. Furthermore, a set of Reg Block mutation operators is designed to evolve the population through all three stages.

II. Related Works

Manually designing high-performance network architectures is highly challenging, requiring considerable domain knowledge and consuming substantial time and resources. Compared with the traditional manually-designed approach, NAS algorithms can automatically design diverse and high-performance architectures without professional domain knowledge.

Regarding the development of NAS algorithms, most works have focused only on improving the classification performance to outperform manually-designed algorithms. For example, the Genetic CNN [20] achieves a 27.87%

top-1 recognition error rate on the ILSVRC2012 dataset [21], which is better than the error rates of most manually-designed network architectures [1], [3], [22]. However, the Genetic CNN also has the most parameters, reaching 156M parameters. Large-scale Evolution [14] achieves good classification accuracy on the CIFAR-100 dataset but requires 40.4M parameters, which is more than those of other network architectures. In addition, while AS-NAS [23] outperforms most NAS algorithms in terms of classification performance, AS-NAS requires a substantial number of parameters. The network architectures obtained by the above algorithms achieve good performance in classification tasks. However, these networks typically involve a large number of parameters, which makes these network architectures difficult to calculate and deploy.

In recent years, some NAS algorithms have focused on multiobjective NAS to construct effective network architectures for real-world applications requiring small size and high accuracy. Multiobjective NAS algorithms aim to optimize their accuracy and consider their resource consumption levels. One kind of multiobjective NAS algorithm, such as ProxlessNAS [24] and FBNet [25], rely on a scalarized objective. This kind of algorithm simultaneously encourages good classification performance while penalizing other objectives. Gradient-based approaches construct a regularization loss to control the trade-off between accuracy and latency. The algorithms in this category need to define preference weight values according to the priority levels of different objectives before searching; this requires a considerable number of trials, consuming substantial time and computational resources. Another category of algorithms includes heuristic algorithms based on multiobjective optimization, such as NSGA-Net [17], MSuNAS [26], and LEMONADE [18]. The algorithms in this category seek high-performance network architectures by simultaneously moving objectives to approximate the Pareto frontier. However, they still

require considerable time and computational resources. The high demands rise due to the training and validation of many network architectures, which are the main computational bottlenecks regarding NAS algorithm development.

NAS algorithms developed to accelerate automated discovery, such as parameter-sharing-based algorithms [27], [28], [29] and single-path sampling-based algorithms [6], [30], have recently attracted much attention. A one-shot supernet was constructed first. The weights of the subnetworks found during the search were all inherited from the supernet, enabling the subnetworks to obtain their weights without training. However, the supernet training process requires considerable time, and supernet optimization is difficult. The algorithms that rely on supernet to completely replace the actual subnetwork weight optimizations are unreliable. Several studies like [31], [32] have proven that the correlation between subnetwork performance obtained by the above algorithms and the test accuracy obtained by gradient descent calculation are both weak. In addition, some algorithms use surrogate models to predict the network architecture accuracies rather than directly training network architectures to reduce computational consumption; examples include PNAS [33], OnceForAll [34], and SemiNAS [35]. The accuracy of the network architecture is predicted by constructing a predictor that can achieve a more accurate network architecture performance evaluation. However, the rank order between the true and predicted accuracy is low. Training surrogate models also requires numerous data samples. The 2k~50k network architectures need to be collected as the training data to train the predictors, which means that the cost of well-trained predictors is exceptionally high.

Instead of evaluating network architecture performance through training, NAS algorithms that do not involve training have been proposed. Several training-free metrics are correlated with the network test accuracy, and these metrics can be directly employed as proxies

for training-based metrics in optimization problems. These training-free metrics are built differently for various NAS methods. Mellor et al. [36] recently studied the overlap between the data point activations in untrained network architectures. They proposed a Jacobian based on input and output data as a training-free metric. However, the principle of this measure has not been explicitly proven or explained. TE-NAS [37] leverages the NTK proposed by Jacot [38] and Hanin [39] to measure the network architecture trainability. In TE-NAS, a strong correlation between the NTK condition number and the network architecture test accuracy was found, displaying good generalization in different search spaces.

In general, NAS algorithms can be roughly divided into reinforcement learning (RL) approaches [28], [40], [41] and EA-based approaches [42], [43], [44]. Real et al. [7] provided a large-scale comparison between EA-based and RL-based algorithms, proving that EA-based methods converge faster than RL-based approaches in the same search space. Moreover, the experimental results in [45] showed that EA-based designs often require fewer computational resources than RL-based designs. EA-based methods considers constructing the network architecture

as a combinatorial optimization problem through EA, with individuals in the population representing network architectures. During evolution, the network architectures pass through crossover and mutation operators, and the individuals with the best fitness are selected as the optimal solution at the end of the evolution.

III. Methods

The goal of LoNAS is to search for highly accurate neural network architectures with few parameters, using little search time and computational resources. An overview of the LoNAS algorithm is presented first in this section. Then, the details of LoNAS are introduced, including the Reg Block design, an expanded search space based on a variable-architecture encoding strategy, a fitness evaluation of individuals based on a training-free proxy (i.e., the NTK strategy), a three-stage EA based on multiple-criteria environmental selection, and a set of mutation operators for Reg Block.

A. Algorithm Overview

An overview of the proposed LoNAS approach is summarized in Algorithm 1. LoNAS starts with a population initialized with random individuals (Line 1).

Algorithm 1. Overview of LoNAS

Input: Reg Block parameters, the population size N , the maximum number of evolution rounds Max_gen , the number of offspring t , and the two separation points in the three-stage evolution process (G_1 and G_2).

Output: The best discovered architecture.

```

1:  $P \leftarrow$  Initialize a population with a size of  $N$  by using the encoding strategy;
2: Evaluate the fitness of the individuals in  $P$  based on the NTK;
3:  $i \leftarrow 1$ 
4: while  $i \leq Max\_gen$  do
5:    $parents \leftarrow$  Select  $t$  parent individuals in  $P$  based on tournament selection;
6:    $Q \leftarrow$  Generate  $t$  offspring individuals from  $parents$  with mutation operators;
7:   if  $i \leq G_1$  or  $G_2 \leq i$  then
8:      $P \leftarrow$  Select  $N$  individuals in  $P \cup Q$  by environmental selection based on the
       NTK;
9:   else
10:     $P \leftarrow$  Select  $N$  individuals in  $P \cup Q$  by environmental selection based on the
      individual lifespan values;
11:   end if
12:    $i \leftarrow i + 1$ 
13: end while
14: return The architecture with the best test accuracy.
```

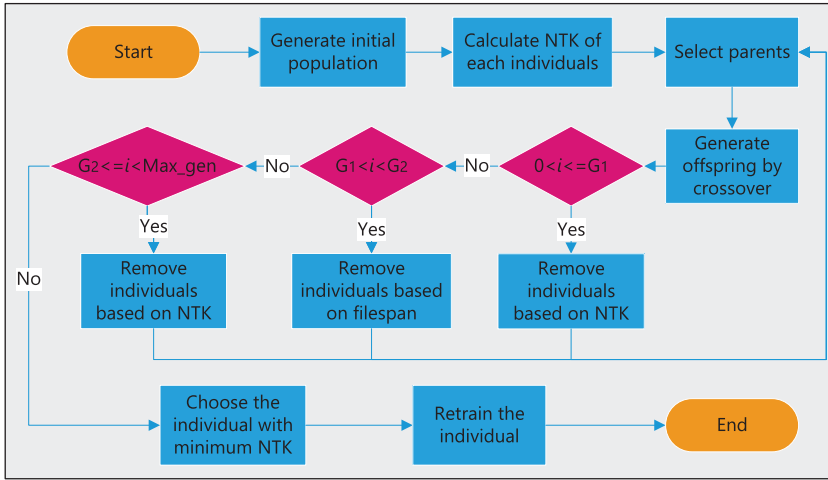


FIGURE 1. The flowchart of the LoNAS algorithm, where i is the evolution round, Max_gen is the maximum number of evolution rounds, and G_1 and G_2 are the two separation points in the three-stage evolution process.

Each individual represents a network architecture encoded based on the variable-architecture encoding strategy. Various network architectures can be constructed by searching for different parameter combinations, which helps to expand the search space. The condition number of the NTK (K_N) is set as the training-free proxy to evaluate individuals. The evaluation time is significantly reduced since complete network training is avoided (Line 2). The iterative evolution process is divided into three stages according to the different environmental selection criteria. The first and third stages use K_N as the criterion to implement environmental selection, and the

second stage leverages the individual lifespan $Lifespan$ as the criterion. Several individuals in the population are selected as parents based on tournament selection (Line 5). The offspring are generated from the parents through mutation operators (Line 6). After the offspring are constructed, they are evaluated and added to the population once constructed. The corresponding criterion is used for environmental selection according to the stage of the current evolution round. Some individuals are eliminated (Line 8 and Line 10) to ensure that the number of surviving individuals in the population is consistent with that in the initial population. Then, a new population

formed by the remaining individuals enters into the next round. The evolution process continues until the predefined maximum number of evolution rounds is reached. Finally, the best network architecture is decoded from the individual in the final population with the best fitness. The corresponding flowchart is shown in Fig. 1.

B. Expanded Search Space Based on the Reg Block

The network architectures in LoNAS are constructed by a variable-architecture encoding strategy based on a novel network block called the Reg Block. A well-designed search space containing numerous potential network architectures is crucial for NAS. Therefore, five dimensions are included to construct the LoNAS search space: the length of the network architecture, the number of group convolutions, the width of each group convolution, the stride of the pooling layers, and the SENet module. Compared with the search spaces with fewer dimensions in [8], [9], [46], the search space is expanded in LoNAS since different network architectures can be constructed in more flexible and fine ways. More potential network architectures can be searched, which leads to better solutions. An overview of the search space and the encoding strategy is shown in Fig. 2.

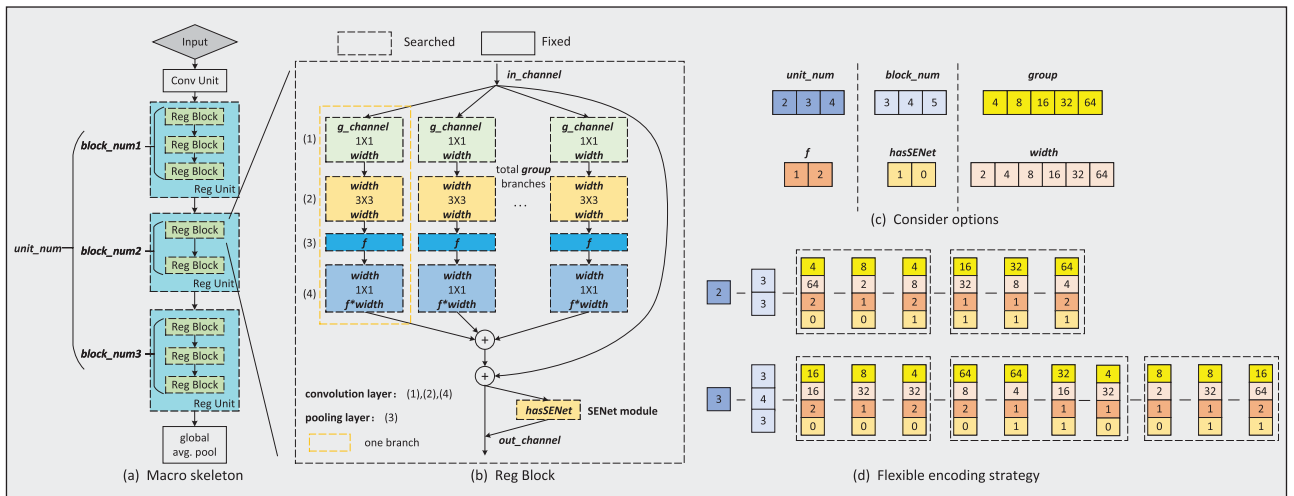


FIGURE 2. LoNAS search space and the encoding strategy. **(a):** The macro skeleton of the network architecture. **(b):** The topology of the Reg Block. **(c):** The considered options for the network architecture parameters. **(d):** A flexible encoding strategy based on the Reg Block.

The novel network block in LoNAS, called Reg Block, is composed of the group convolution and SENet module and is used to reduce the number of parameters and improve the classification performance.

1) Network Architecture Macro Skeletons

The proposed search space follows a block-based design [8], [9], [26], in which various network architectures are flexibly constructed according to different parameter options, as shown in Fig. 2(a). The macro skeleton of each network architecture is initialized with a Conv Unit to extract the input data features; this unit contains one 1×1 convolutional layer with 64 output channels and a batch normalization

layer [47]. The main body of the macro skeleton includes *unit_num* Reg Units. Each Reg Unit is composed of *block_num* Reg Blocks. Each Reg Block is generated randomly based on a set of parameters that can be searched automatically. A global average pooling layer is placed at the end of the skeleton to flatten the feature maps into a feature vector. A fully connected layer with a softmax layer is set as the classification module to transform the feature vector into the final prediction result. The

network architecture construction process in the LoNAS search space is summarized in Algorithm 2.

2) Reg Block

The standard convolution [2] achieves good classification performance but requires many parameters, which is not conducive to designing highly accurate network architectures with few parameters. In this paper, a novel network block called the Reg Block is designed. The Reg Block consists of group convolution [22] and SENet module [48]. The group convolution reduces the number of parameters, and the SENet module improves the classification performance. The Reg Block topology is shown in Fig. 2(b).

In the Reg Block, the input features are partitioned into a certain number of groups, decomposing the standard convolution operation into multiple independent convolution branches. Compared with the standard convolution operation, group convolution reduces the number of parameters without remarkably reducing the network classification performance. Based on [22], the group convolution is improved by adding three convolutional layers and one pooling layer in each branch, which allows the network to extract more valuable features. The first and fourth convolutional layers use 1×1 kernels to adjust the number of feature maps, and the second convolution layer uses 3×3 kernels to extract feature maps. All convolutional layers follow a sequence, including a convolution operation, a ReLU [49], and a batch normalization process. The pooling layer halves the input data size and maintains the features invariant, satisfying the calculation constraint. For $M \times M$ input data, the number of pooling layers used to halve the data size cannot be larger than $\lfloor \log_2(M) \rfloor$, or the size of the input data is reduced to less than 1. The Reg Block output is formed by concatenating the output features of each branch, the residual connections, and an SENet module [48]. The SENet module simulates an attention mechanism [50], [51] through *squeeze-and-excitation*, which ensures that the network architecture focuses on the

Algorithm 2. Network architecture construction in the LoNAS search space

Input: The size of the input data $M \times M$, *unit_num* list, *block_num* list, *group* list, *width* list, *f*.

Output: The generated network architecture.

```

1:  $d \leftarrow$  Calculate the maximum number of pooling layers with a stride of 2 by  $\lfloor \log_2(M) \rfloor$ ;
2:  $q \leftarrow$  An empty queue;
3: Construct a Conv Unit and add it to  $q$ ;
4: unit_num  $\leftarrow$  Randomly select a value in the unit_num list;
5: for  $i \leftarrow 1$  to unit_num do
6:   block_num  $\leftarrow$  Randomly select a value in the block_num list;
7:   for  $j \leftarrow 1$  to block_num do
8:     group, width  $\leftarrow$  Randomly select values in the group and width lists;
9:     while group  $\times$  width does not meet the constraint do
10:      group, width  $\leftarrow$  Randomly reselect group and width;
11:   end while
12:   if The number of used pooling layers with a stride of 2 is less than  $d$  then
13:      $f \leftarrow$  Randomly select a value in  $\{1, 2\}$ ;
14:   else
15:      $f \leftarrow 1$ ;
16:   end if
17:    $r \leftarrow$  Uniformly generate a value in  $[0, 1]$ ;
18:   if  $0.5 \leq r$  then
19:     hasSENet  $\leftarrow 1$ ;
20:   else
21:     hasSENet  $\leftarrow 0$ ;
22:   end if
23:   block  $\leftarrow$  Generate a Reg Block according to the parameters group, width,  $f$ , and hasSENet;
24:   Add the block to  $q$ ;
25: end for
26: end for
27:  $q \leftarrow$  Construct a global average pooling layer and add it to  $q$ ;
28: Generate a network architecture according to  $q$ ;
29: return A network architecture

```

NTK can be used to characterize the trainability of each network architecture and therefore is designed to evaluate the performance of individuals in the population that can significantly save search time and computational resources.

most informative components of the features, thereby improving the representational capacity of the network architecture.

3) Variable-Architecture Encoding Strategy

A variable-architecture encoding strategy based on the Reg Block is designed to construct a large search space with numerous network architectures, as shown in Fig. 2(c) and (d). Searching different parameter combinations allows various network architectures to be flexibly constructed. First, the value of the *unit_num* parameter is randomly selected to determine the initial number of Reg Units. Then, in each Reg Unit, *block_num* Reg Blocks are randomly generated.

As depicted in Fig. 2(b), the input of each Reg Block comes from the output of the previous Reg Block. The input feature maps are divided into *group* branches. Each branch has *g_channel* channels, which are determined by *in_channel/group*. The width of the bottleneck of each branch is set as the *width*. The stride of the pooling layer is denoted as *f*. *f* is set to 2 when the feature map is downsampled; otherwise, *f* is set to 1. At the end of each Reg Block, an SENet module is added with a 50% probability, which improves the representational ability of the network without introducing too many parameters. This dimensionality is denoted by the parameter *hasSENet*. The value of *hasSENet* is set to either 1 or 0, representing whether an SENet module is in the Reg Block. When generating a Reg Block, the parameters *group* and *width* are randomly selected from two specified lists. The parameter *f* is randomly selected from {1, 2}. When the number of existing pooling layers performing the halving meets the preset constraint, *f* is set to 1.

These architectural parameters are automatically searched and encoded as a digital string.

By including the group convolution and SENet module, more dimensions for constructing network architectures are added, effectively expanding the search space and allowing more networks with better performance to be obtained.

4) Computational Restriction

The length of the network architecture has an important effect on the test accuracy [1]. Longer network architectures generally obtain better test accuracy but require more parameters, which makes these network architectures more challenging to calculate. Through careful structure design, some short network architectures can achieve good classification performance [2], [4], and determining these architectures is the goal of the algorithm proposed in this paper. Although the lengths of the network architectures in the LoNAS search space can vary according to the encoding strategy, the lengths are restricted to ensure that the number of parameters of each network architecture in the search space is limited. Network architectures with too many or too few parameters can be avoided since they are not the target, thereby improving the search efficiency. The number of convolutional layers determines the length of the network architecture:

$$Length = 1 + \sum_{i=U_{min}}^{U_{max}} 3 \cdot U_i^{block} \quad (1)$$

where *unit_num* is limited to the range $[U_{min}, U_{max}]$. U_i^{block} represents the number of Reg Blocks in the *i*th Reg Unit, which corresponds to the parameter *block_num*. *block_num* is limited to the range $[block_{min}, block_{max}]$. The number

of Reg Blocks in each Reg Unit must be multiplied by a factor of 3 since each Reg Block contains three convolutional layers. The total length of the network architecture is determined by adding a convolutional layer in the Conv Unit.

The number of convolutional kernels also affects the number of network parameters. When the convolutional kernels are the same size, more convolutional kernels improve the feature extraction ability, leading to better classification performance for the network architecture while requiring more parameters. Therefore, the number of convolutional kernels should also be limited to avoid requiring too many parameters. The number of convolutional kernels reflects the number of output feature maps in the intermediate layer, which is determined by the product of the *group* and *width* parameters. The proposed encoding strategy implements two rules to restrict the number of feature maps. The first rule is that larger values have lower selection probabilities when randomly selecting the *group* and *width* parameters. The other rule is that maximum and minimum values are set for the product of *group* and *width*. If $group \times width$ is larger than the maximum value or less than the minimum value, both parameters need to be randomly selected again until the product meets the limit. Algorithm 2 shows the application of these rules in the construction of the network architecture. Under these rules, the overall number of network architecture parameters in the search space is limited, which guarantees better search performance to discover highly accurate network architectures with few parameters.

Compared to the coding strategies in [9], [26], [46], there are more parameter dimensions included in the proposed encoding strategy. As a result, finer blocks can be generated, and more various network architectures can be discovered. In addition, network architectures with too few or too many parameters that do not meet the parameter requirements are not generated by adding computational restrictions, resulting in a

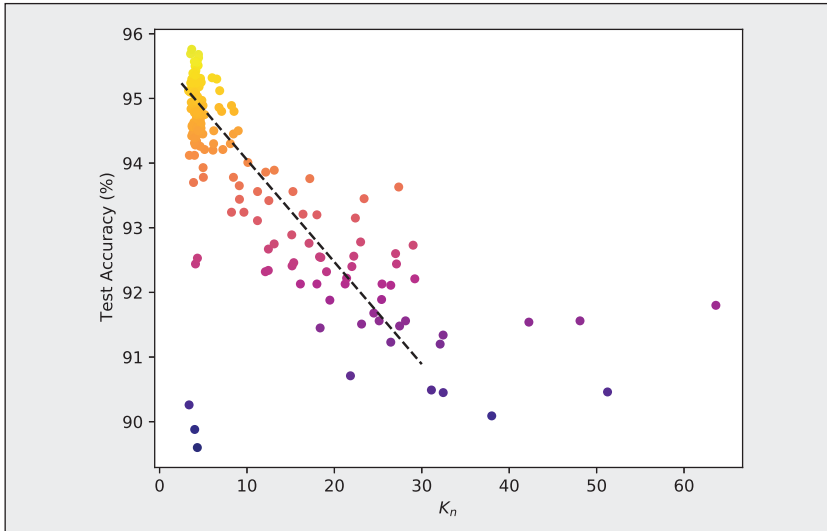


FIGURE 3. The negative correlation between K_N and the architecture test accuracy in the LoNAS search space on CIFAR-10. Dots in lighter colors represent more accurate network architectures. The Kendall-tau correlation is -0.186 .

more efficient search than other coding strategies.

When the population is initialized, the parameters *unit_num*, *block_num*, *group*, *width*, and *f* are randomly selected from the specified parameter settings to generate individuals. More details on the parameter settings are discussed in Section IV.

C. Fitness of Individuals Based on a Training-Free Proxy

Since training network architectures through backpropagation requires

considerable search time in NAS, a training-free proxy is proposed to accelerate the search procedure. In this paper, the NTK is designed as the proxy and set as the individual fitness. The NTK can be used to characterize the trainability of each network architecture. A higher trainability represents a network architecture with superior accuracy [52]. According to [37], the NTK can be formulated as

$$\mu_t(X_{train}) = (I - e^{-\eta\lambda_i t})Y_{train} \quad (2)$$

where $\mu_t(x)$ denotes the network outputs, λ_i represents the eigenvalues of

the NTK between the training inputs, X_{train} and Y_{train} are drawn from the training set, t is the training step, and η is the learning rate scale. The eigenvalues are ordered $\lambda_0 \geq \dots \geq \lambda_m$, and $K_N = \lambda_0/\lambda_m$ is set as the number of conditions. m is the number of NTK eigenvalues. The network becomes untrainable when K_N tends to be larger. K_N can be calculated without gradient descent and is used to characterize the trainability of the network.

In this paper, K_N is used as a proxy to evaluate the classification performance of different networks. This approach significantly reduces the evaluation time since the networks are not directly trained. In an EA, evaluating individuals usually requires considerable time. Therefore, K_N is applied to describe an individual's fitness during evolution, accelerating the fitness evaluation in the EA. Fig. 3 shows the correlation between K_N and the test accuracy of the network architectures among 200 individuals that were randomly generated in the LoNAS search space. Fig. 3 shows that K_N is negatively correlated with the test accuracy. During the evolution process, minimizing K_N can help find network architectures with high accuracy. The application of a training-free proxy saves considerable search time and computational resources.

Fig. 3 shows some fluctuations in the negative correlation. To study the fluctuations between K_N and the test accuracy, K_N values of 11 different networks are collected, as shown in Fig. 4. K_N is calculated 10 times for each network architecture. For the same network architecture, every time the input training data are randomly selected with the same batch size, the K_N result changes within a range since K_N is related to the input data according to (2). A network architecture with a test accuracy of 91.1% is taken as an example and plotted as the dark dots in Fig. 4. The K_N value varies from 61 to 84. The following calculation method is applied to reduce the impact of this deviation. For each individual, K_N is calculated with 12 independent runs. The highest and lowest values are removed, and the average of the remaining 10 values is set as the final

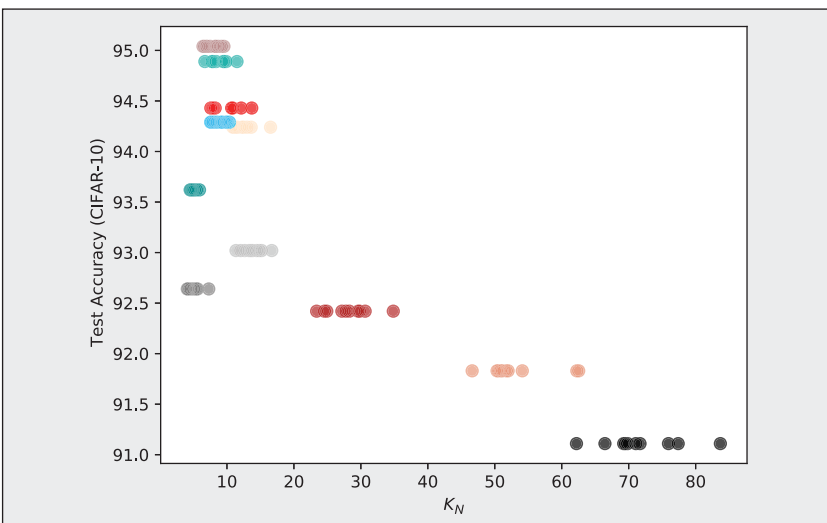


FIGURE 4. Fluctuations in the correlation between K_N and the test accuracy of the network architecture in LoNAS on CIFAR-10. The 11 colored dots represent 11 different network architectures. The K_N of each network with fixed test accuracy is independently calculated 10 times with random training data on CIFAR-10.

The three-stage EA can effectively balance exploration and exploitation by the environmental selection process criteria based on the K_N and individual lifespans.

$$Criterion = \begin{cases} K_N & \text{if } 0 < G \leq G_1 \text{ or } G_2 < G \leq Max_gen \\ Lifespan & \text{if } G_1 < G \leq G_2 \end{cases} \quad (3)$$

K_N result. In addition, performing multiple calculations ensures that the network receives most input data rather than a small part of the data, which improves the generalization of the network to the input data. In [37], K_N was combined with another indicator calculated by normalization. In this paper, only K_N is applied to rank the overall network architectures, as this approach is simpler and thus accelerates the search process.

D. Three-Stage EA Based on Multiple-Criteria Environmental Selection

To balance exploration and exploitation, a three-stage EA is proposed based on a multiple-criteria environmental selection strategy, which differs from other three-stage EA methods [53], [54] with multiple mutation strategies. The two criteria are K_N and the *Lifespan* [7], which indicates the number of evolution rounds an individual has experienced.

1) Process of the Three-Stage EA

The environmental selection criteria are determined as follows:

where G_1 and G_2 are the round numbers used to divide the evolution process, G represents the current round, and Max_gen is the maximum number

of evolution rounds. Fig. 5 shows the whole evolution process of the three-stage EA.

In the first stage ($0 < G \leq G_1$) and third stage ($G_2 < G \leq Max_gen$), K_N is applied as the environmental selection criterion. In the second stage ($G_1 < G \leq G_2$), the individual's *Lifespan* is used as the environmental selection criterion. The evolution starts after the population is initialized with n individuals. First, k individuals are randomly selected. From these k individuals, t individuals with the best fitness are sampled as the parents. t offspring individuals are constructed from the parents according to the set of mutation operators described in Section III-E. Once the offspring individuals are constructed, they are evaluated and added to the existing population, resulting in $t + n$ individuals. Then, according to the stage of the current evolutionary round, the corresponding criterion is used for environmental selection, eliminating the t worst individuals. The remaining n individuals form a new population that enters the next evolution round.

2) Advantages of the Three-Stage EA

The search process of traditional EA approaches is easily trapped in local optima since most offspring inherit only

some of the good parents during evolution process [7]. In aging evolution [7], individuals are discarded according to their lifespans. During evolution, older individuals with good fitness continue to be discarded. These individuals are removed as potential optimal solutions from the search space, which slows the population's convergence, causing convergence instability.

Comprehensively considering the traditional EA and aging evolution process, a three-stage EA is proposed in this paper. In the first and third stages of evolution, individuals with smaller K_N values are retained during the selection process. In the second stage, younger individuals in the population are saved. During the first stage, outstanding individuals are selected to enter the later evolutionary process to ensure that offspring can inherit from these individuals, improving the overall performance of the population and ensuring that sufficient potential optimal solutions are contained in the population. Then, in the second stage, the population is frequently renewed, enabling more exploration of the search space and increasing the diversity of individuals. Moreover, a limited number of rounds are performed to ensure that not all good individuals are discarded. Finally, in the third stage, outstanding individuals are retained during the environmental selection process, leading the population to converge to the optimal solution, which helps ensure exploitation. The experiments conducted in Section IV study the effectiveness of the multiple-criteria environmental selection strategy.

E. Mutation Operators for the Block-Based Network Architecture

The offspring individuals in the population are generated from mutation operations. In this paper, mutation operators are performed only in the Reg Unit, while the Conv Unit is not involved due to its specific function. For the mutation operators, a mutation position pos_{ij} , which represents the position of the j th Reg Block in the i th Reg Unit, is randomly selected according to the length of the parent individuals. The Reg Unit and Reg Block positions determine the

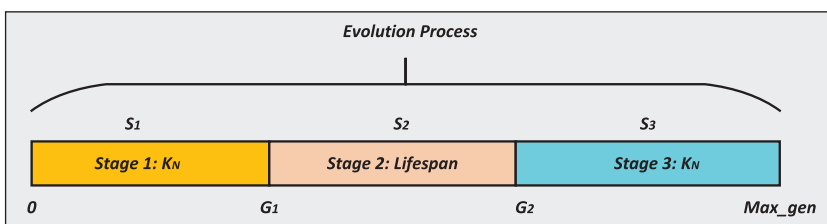


FIGURE 5. The whole evolution process of the three-stage EA according to the different environmental selection criteria.

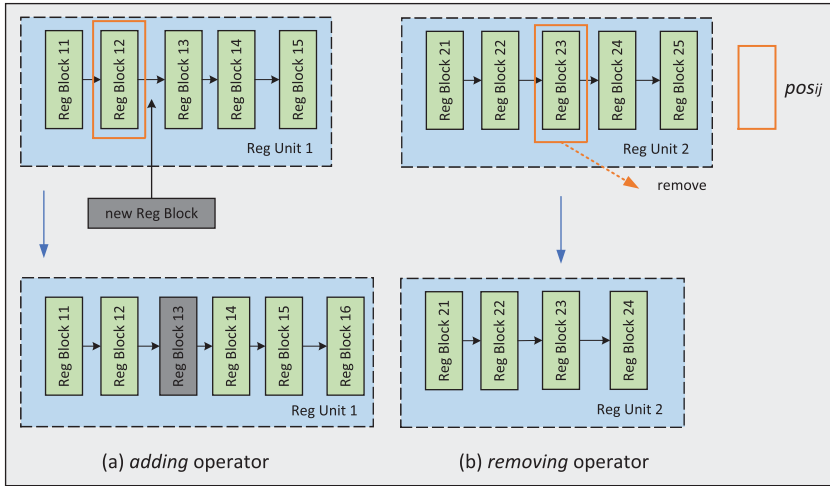


FIGURE 6. Examples of the *adding* and *removing* mutation operators.

mutation position. Then, one of the mutation operators is randomly selected to carry out the parent individuals. According to the block-based network architecture, the designed mutation operators are as follows:

- ❑ *adding*: Adding a Reg Block with random settings;
- ❑ *removing*: Removing the Reg Block at the selected position;
- ❑ *altering*: Randomly altering the parameter values of the Reg Block at the selected position.

More specifically, the *adding* operator generates a Reg Block with random parameters, which is inserted after position pos_{ij} . The *removing* operator deletes the Reg Block at position pos_{ij} . The *altering* operator randomly generates a new set of parameters to replace the old parameters of the Reg Block at position pos_{ij} . Fig. 6 shows examples of adding a new Reg Block and removing an existing Reg Block. In Fig. 6(a), a new Reg Block is randomly generated and inserted after Reg Block 11. In Fig. 6(b), Reg Block 23 is removed from Reg Unit 2. Notably, the length of the parent individual must be considered when implementing the *adding* and *removing* operators. If the length reaches the upper limit, the *adding* operator cannot be performed, and only the other two types of operators can be selected. When the length of the original individual reaches the lower limit, the *removing* operator cannot be implemented. In addition, when the *altering* operator is selected, the newly generated

parameters need to meet the restrictions regarding the number of feature maps.

IV. Experiments

To investigate the effectiveness of the proposed algorithm, three benchmark datasets, CIFAR-10, CIFAR-100, and ImageNet-16-120, are adopted to evaluate the network architectures. The experimental results of the proposed algorithm are compared with those of different state-of-the-art NAS algorithms.

A. Implementation Details

1) Benchmark Datasets

CIFAR-10 is a 10-class classification dataset containing 50K training images and 10K testing images. CIFAR-100 is similar to CIFAR-10 but contains 100 classification categories. The training datasets are split (80%/20%) to generate training and validation datasets for ablation experiments. The testing dataset is used only to determine the final test accuracies of the network architectures. The datasets are augmented before training. In this work, the same augmentation techniques that are usually applied in other algorithms, such as random cropping and random horizontal flipping [2], [4], are employed to ensure a fair comparison. In addition, the commonly used data preprocessing technique *cutout* [55] is applied in this paper.

2) Hyperparameters of LoNAS

The number of Reg Units $unit_num$ is set to {2, 3, 4}. The number of Reg Blocks

contained in one Reg Unit $block_num$ is set to {3, 4}. Thus, the network length ranges from [19-49] according to (1). The available values for the $group$ list in the Reg Blocks are set to {2, 4, 8, 16, 32, 64}, and the $width$ list is set to {4, 8, 16, 32, 64}. The maximum and minimum numbers of network feature maps are 1024 and 64, respectively. To ensure that the constructed network architectures have limited numbers of parameters, the larger values in the $group$ and $width$ lists have lower selection probabilities. For the $group$ list, the probabilities of 2, 4, 6, and 16 are set to 0.19, and the probabilities of 32 and 64 are set to 0.12. For the $width$ list, the probabilities of 4, 8, and 16 are set to 0.24, while the probabilities of 32 and 64 are set to 0.14. During the search process, the population size, the maximum number of evolutions, and the batch size for calculating K_N are set to 40, 50, and 32, respectively. The values of k and t in the tournament selection process are set to 5 and 2, respectively. To achieve better evolutionary performance, each stage in the environmental selection process is set to a different length in the experiments, as shown in Section IV-C. The best individuals in the final population are selected for training.

3) Network Training

During training, stochastic gradient descent (SGD) with a momentum rate of 0.9 and an L2 weight decay of 5×10^{-4} is used to conduct training for 300 epochs. The learning rate is initialized to 0.1 with a cosine decay schedule. Then, the network architecture with the best test accuracy is selected and retrained independently 10 times with the same routine. All the experiments are performed on a single NVIDIA 1080Ti GPU.

B. Performance Investigation of the Reg Block

Two ablation experiments are conducted on CIFAR-10 dataset to verify the effectiveness of the Reg Block. The first experiment is performed to verify the effectiveness of the group convolution, and the second experiment is used

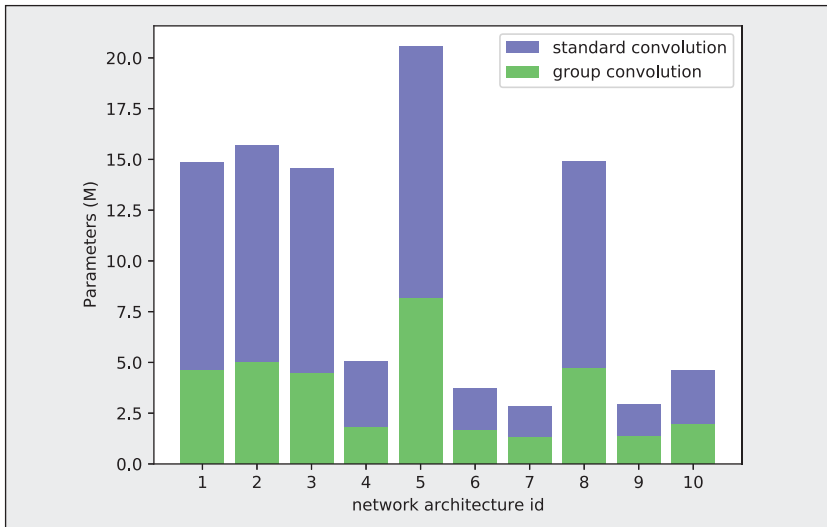


FIGURE 7. Comparison of the number of parameters between the group convolution and the standard convolution.

to investigate the effectiveness of the SENet module. Ten individuals in the final population are randomly selected

for both experiments, and all of these individuals contain group convolution and SENet modules.

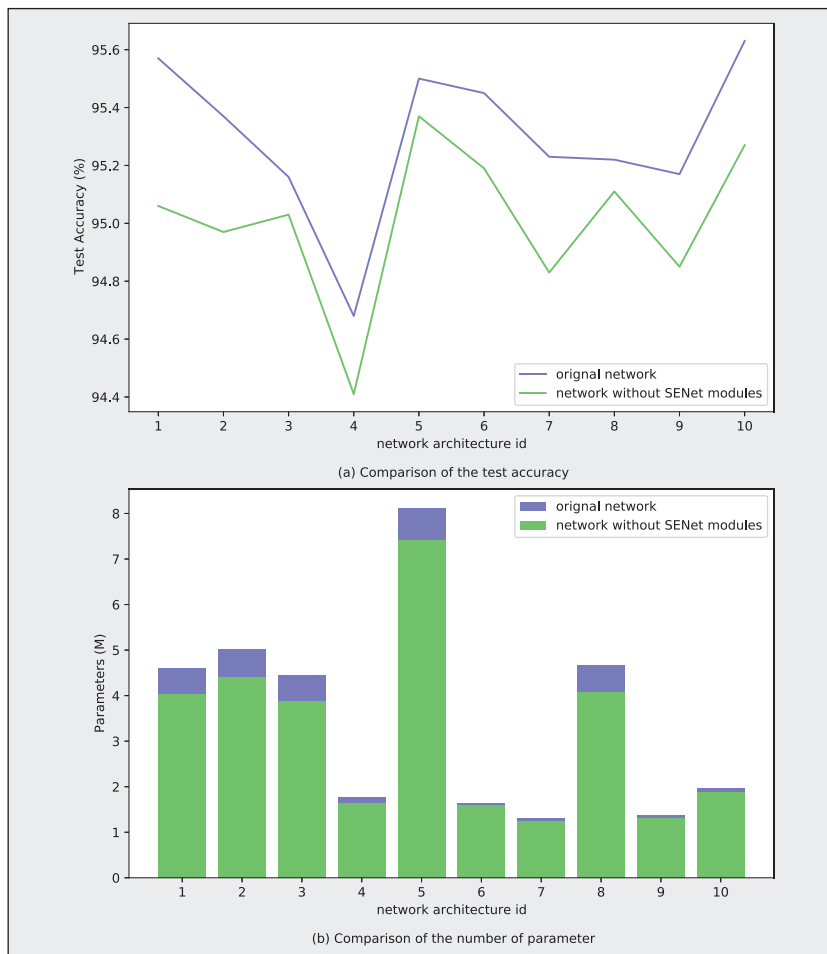


FIGURE 8. Performance comparison between original network architectures and those without SENet modules.

The first ablation experiment was performed to verify the impact of the group convolution on the number of parameters. The number of parameters of each individual is recorded first. Then, the group convolution of each individual is transformed into a standard convolution while keeping the remaining topology the same, and the corresponding number of parameters is recorded. The comparison results are shown in Fig. 7. The green bar represents the group convolution results, and the blue bar represents the standard convolution results. Fig. 7 shows that the group convolution consumes fewer parameters than the standard convolution, and each individual with the group convolution has approximately half the number of parameters. Therefore, the group convolution effectively reduces the number of parameters.

In the second ablation experiment, the effectiveness of the SENet module in improving the test accuracy and reducing the number of parameters is verified. The test accuracy and number of parameters of each individual are recorded through 10 independent trials. Then, all SENet modules of each individual are removed, and the test accuracy and number of parameters are recorded. The comparison results regarding the test accuracy and the number of parameters are shown in Fig. 8(a) and (b), respectively. In Fig. 8, the blue line and the blue bar represent the original network architectures, and the green line and the green bar represent the architectures without SENet modules. Fig. 8(a) shows that compared to the original network architectures, the accuracy of the networks without the SENet modules is substantially reduced, indicating that the SENet module improves the accuracy. Fig. 8(b) indicates that the addition of SENet module only slightly increases the number of parameters. These results demonstrate that the SENet module remarkably improves the classification performance of the network architecture while adding only a moderate number of parameters.

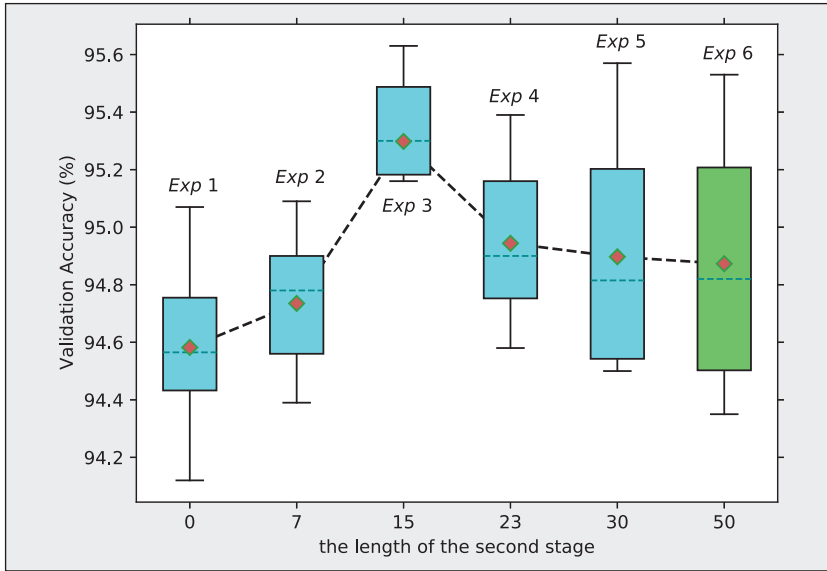


FIGURE 9. The impact of the length of the second stage on the three-stage EA.

C. The Effectiveness of the Three-Stage EA and the Best Length Division for the Three Stages

Six independent experiments are conducted to investigate the effectiveness of the three-stage EA. In each experiment, an EA with the same number of evolution rounds and different lengths for the three stages is implemented. The classification performance of each final population is recorded. Five experiments, named *Exp 1* ~ *5*, have the same fixed length for the first stage but different lengths for the second and third stages to

investigate the effects of the three-stage length on the population. Another experiment named *Exp 6* only goes through the second stage. In *Exp 1* ~ *5*, the value of G_2 is changed to investigate the impact of different stage lengths on the validation accuracy of the final population. The length of the second stage is recorded. In Fig. 9, each box represents the overall validation accuracy of a population, the length of the box represents the accuracy deviations between individuals, and the dot and dotted line in the box represent the average and

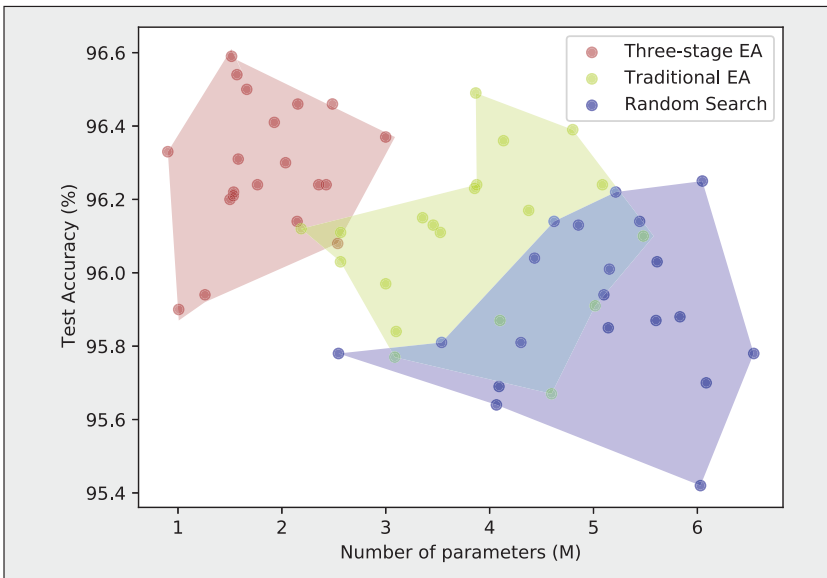


FIGURE 10. Comparison of the three-stage EA, traditional EA, and random search method in terms of the test accuracy and the number of parameters.

median accuracy values, respectively. The extended lines at both ends of the box represent the maximum and minimum accuracies. The light blue boxes show the results of *Exp 1* ~ *5*, and the green box represents the results of *Exp 6*.

Among the light blue boxes, when the length of the second stage is set to 0, the EA degenerates to the traditional EA with fixed-criteria environment selection. Fig. 9 clearly shows that the average validation accuracy achieved by the traditional EA is lower than the accuracies of the other three-stage EAs. The second stage helps the population to converge to network architectures with better classification performance since more of the search space is explored.

When the length of the second stage is increased, the population's average accuracy initially increases and then decreases later. When the length of the second stage is set to 50, which is implemented in *Exp 6*, the EA degenerates to an aging evolution process. Fig. 9 shows that the green box and its extended lines have the longest lengths, which indicates the most extensive search instability; it illustrates a prolonged second stage that causes the population to generate more individuals with poor fitness, reducing the number of the optimal solution. The box and extended lines of *Exp 3* have the shortest lengths, indicating that the individuals in this population have the smallest differences. *Exp 3* shows that utilizing a sufficient length for the third stage improves the exploration ability of the algorithm, which helps remove poorly performing individuals and increase the number of optimal solutions. The stability of the search procedure is also improved.

In summary, using an appropriate length for each stage can help to effectively balance exploration and exploitation, leading to better optimal solutions. Based on the ablation experiments, when G_1 and G_2 are set to 15 and 30, respectively, the population shows the best performance. The lengths of the three stages are set to 15, 15, and 20, and the values of $G_1 = 15$ and $G_2 = 30$ are used in the following experiments.

TABLE I Comparison with the state-of-the-art peer competitors in terms of the test accuracy (%), number of parameters (M), search GPU days, and number of GPUs used on the CIFAR-10 and CIFAR-100 datasets. Test accuracy with mean and deviation are reported.

	CIFAR10	CIFAR100	PARAMETERS (M)	SEARCH TIME (GPU DAYS)	GPUs
Manually designed					
FractalNet [56]	94.78	77.7	22.3	-	-
DenseNet ($k = 24$) [4]	96.26	80.75	27.2	-	-
DenseNet-B ($k = 40$) [4]	96.54	82.82	25.6	-	-
Wide ResNet [57]	95.85	79.50	36.5	-	-
ResNeXt-29 (8x64d) [58]	96.35	82.23	34.4	-	8
Semiautomatic					
Hierarchical Evolution [59]	96.37	-	61.3	1.5	200
NASNet-A [13]	97.35	-	3.3	2000	500
DARTS [28]	97.0(0.14)	-	3.3	1.5	1
P-DARTS [60]	97.5(0.12)	-	3.4	0.3	1
PC-DARTS [40]	97.43(0.07)	-	3.6	0.1	1
SP-DARTS [61]	97.29	-	3.64	0.11	1
ENAS(macro) [30]	96.13(0.12)	-	38.0	0.32	1
ENAS(micro) [30]	97.11(0.11)	-	4.6	0.5	1
Block-QNN-S [62]	95.62	79.35	6.1	90	32
TE-NAS [37]	97.37(0.064)	-	3.8	0.05	1
AS-NAS [23]	97.67	-	16.6	-	8
Automatic					
Large-scale Evolution [14]	94.60	-	5.4	2750	-
Large-scale Evolution [14]	-	77.00	40.4	2750	-
NAS [63]	93.99	-	2.5	22400	800
AmoebaNet-A [7]	96.66(0.06)	-	3.2	3150	450
AE-CNN [8]	95.7(0.18)	-	2.0	27	3
AE-CNN [8]	-	79.15(0.18)	5.4	36	3
CNN-GA [9]	96.78	-	2.9	35	3
CNN-GA [9]	-	79.47	4.1	40	3
NSGA-Net [17]	97.5(0.10)	-	26.8	4	1
LF-MOGP [64]	95.87	-	1.07	10	1
LF-MOGP [64]	-	73.63	4.13	13	1
EX-Net(ours)	96.95(0.07)	-	2.0	0.02	1
EX-Net(ours)	-	81.52(0.07)	4.3	0.02	1

Fig. 10 shows the search performance of different search algorithms in terms of the test accuracy and number of parameters. The proposed three-stage EA (brown area) is compared with the traditional EA (green-yellow area) and the random search method (blue area). Each area describes the overall performance of individuals in terms of the test accuracy and the number of parameters. Specifically, the same

population, which contains 20 randomly generated individuals, is used for the three search algorithms, and each algorithm goes through 50 search iterations. In the traditional EA and the random search method, K_N is used to evaluate the performance of the individuals. Fig. 10 shows that individuals discovered by the three-stage EA have higher test accuracy and fewer parameters than those obtained by the traditional EA

and random search method. The overlapping areas between those methods indicate the similarity of the search performance. It can be seen in Fig. 10 that the traditional EA is slightly better than the random search since there are many overlapping areas, which means that many individuals found by the traditional EA have similar performance to those found by the random search. The three-stage EA is much better than the traditional EA since there are few overlapping areas. Furthermore, the populations produced by the three-stage EA have small parameter and test accuracy ranges, indicating that the three-stage EA is robust on the discovered solutions.

D. Overall Results

A set of the state-of-the-art algorithms are used for comparison to comprehensively evaluate the performance of the proposed algorithm. The results are considered in terms of the classification accuracy, number of parameters, search time cost, and required computational resources. The compared algorithms can be roughly divided into three categories. The first category includes manually-designed network architectures. The second category contains semiautomatic NAS. The third category includes automatic NAS algorithms. Table I shows the comparison results. The results of peer competitors are obtained from their published seminal papers. For the proposed algorithm, the best network architecture discovered by LoNAS denoted as EX-Net, is selected from the final population with 10 independent trials.

1) Comparison With Manually-Designed Network Architectures

Comparing the proposed approach with manually-designed state-of-the-art network architectures, the results show that EX-Net achieves considerably better test accuracies on CIFAR-10 and CIFAR-100 with less parameters than FractalNet and Wide ResNet. Compared to DenseNet ($k = 24$), EX-Net achieves better test accuracy on CIFAR-10 and CIFAR-100, and EX-Net uses only 7.3% and 15.8% of the number of parameters in DenseNet ($k = 24$) on CIFAR-10 and CIFAR-100, respectively. Compared to DenseNet-B ($k = 40$) and ResNeXt-29 (8x64d), the test accuracy of EX-Net on CIFAR-10 is

better. On CIFAR-100, although the accuracy of EX-Net is not as high, the number of parameters in EX-Net only consumes 16.8% and 12.5% of the numbers of parameters in DenseNet-B ($k = 40$) and ResNeXt-29 ($8 \times 64d$), respectively, demonstrating a significant reduction. EX-Net uses only 1/8 of the GPU resources consumed by ResNeXt-29 ($8 \times 64d$). In summary, EX-Net achieves higher accuracy than the state-of-the-art manually-designed network architectures and significantly outperforms all competing approaches on CIFAR-10. Additionally, EX-Net employs substantially fewer parameters than these comparison algorithms.

2) Comparison With Semiautomatic NAS Methods

Regarding the semiautomatic NAS algorithms, EX-Net outperforms Hierarchical Evolution, Block-QNN-S, and ENAS (macro) in terms of the test accuracy and number of parameters while significantly reducing the search time cost (16x~4500x reductions). Compared to NASNet-A, EX-Net shows slightly worse test accuracy but requires fewer parameters than NASNet-A. Moreover, EX-Net is 100,000x faster than NASNet-A and consumes only approximately 1/500 of the GPU resources consumed by NASNet-A. DARTS and ENAS (micro) achieve slightly better accuracy on CIFAR-10 than EX-Net, while EX-Net has fewer parameters. DARTS and ENAS (micro) have larger deviations in test accuracy than EX-Net. With the same GPU resource consumption, EX-Net takes 75x and 25x less search time than these two methods. Compared to P-DARTS and PC-DARTS, EX-Net consumes fewer parameters and GPU Days while showing slightly reduced test accuracy. Moreover, EX-Net can achieve more minor deviations across different search rounds. SP-DARTS achieves slightly better accuracy on CIFAR-10 than EX-Net but has nearly 2x more parameters and 5x more search time. In addition, although the accuracy of EX-Net is less than that of TE-NAS, the number of EX-Net parameters and the number of GPU Days consumed by EX-Net are only half of

TABLE II Search time (minutes) on CIFAR-10 and CIFAR-100 by LoNAS.

NO. OF RUNS	1	2	3	4	5	6	7	8	9	10	AVERAGE	VARIANCE
CIFAR-10	28.8	29.5	28.4	28.7	29.4	29.1	28.7	28.3	29.4	28.8	28.91	0.16
CIFAR-100	28.7	28.6	29.4	29.3	29.1	28.6	28.9	29.4	29.3	29.4	29.07	0.10

TABLE III The best architecture on CIFAR-10 by comprehensively considering the test accuracy and the number of parameters of the network architectures.

TYPE	BLOCK	CONFIGURATION
Conv Unit	-	input size=32*32, input channel=3, output channel=64
Reg Unit1	Reg Block1	group=16, width=8, hasSENet=1, f=1
	Reg Block2	group=16, width=32, hasSENet=1, f=1
	Reg Block3	group=8, width=8, hasSENet=1, f=2
	Reg Block4	group=64, width=16, hasSENet=1, f=2
Reg Unit2	Reg Block5	group=8, width=8, hasSENet=0, f=1
	Reg Block6	group=16, width=8, hasSENet=1, f=1
	Reg Block7	group=32, width=8, hasSENet=1, f=1
Reg Unit3	Reg Block8	group=4, width=16, hasSENet=1, f=1
	Reg Block9	group=64, width=8, hasSENet=1, f=2
	Reg Block10	group=32, width=4, hasSENet=1, f=1
	Reg Block11	group=32, width=16, hasSENet=0, f=1

TABLE IV The best architecture on CIFAR-100 by comprehensively considering the test accuracy and the number of parameters of the network architectures.

TYPE	BLOCK	CONFIGURATION
Conv Unit	-	input size=32*32, input channel=3, output channel=64
Reg Unit1	Reg Block1	group=8, width=32, hasSENet=1, f=1
	Reg Block2	group=16, width=4, hasSENet=1, f=1
	Reg Block3	group=4, width=32, hasSENet=0, f=1
Reg Unit2	Reg Block4	group=32, width=8, hasSENet=1, f=2
	Reg Block5	group=16, width=16, hasSENet=1, f=1
	Reg Block6	group=16, width=4, hasSENet=0, f=1
	Reg Block7	group=16, width=4, hasSENet=1, f=1
Reg Unit3	Reg Block8	group=4, width=16, hasSENet=0, f=1
	Reg Block9	group=4, width=16, hasSENet=0, f=1
	Reg Block10	group=16, width=16, hasSENet=0, f=1
	Reg Block11	group=64, width=4, hasSENet=1, f=1
Reg Unit4	Reg Block12	group=16, width=8, hasSENet=1, f=2
	Reg Block13	group=16, width=8, hasSENet=1, f=1
	Reg Block14	group=16, width=16, hasSENet=1, f=1
	Reg Block15	group=32, width=16, hasSENet=1, f=1

those required by TE-NAS. Compared to AS-NAS, EX-Net shows lower test accuracy. However, EX-Net consumes nearly 9x fewer parameters and requires

only 8x fewer computational resources than AS-NAS and is thus competitive algorithm. Therefore, compared to the semiautomatic NAS algorithms, EX-Net

TABLE V Comparison of different network architectures on ImageNet-16-120.

ARCHITECTURE	TEST ACCURACY(%)	PARAMETERS(M)	GPU DAYS	GPUs
ENAS [30]	16.32	4.6	0.5	1
AmoebaNet-A [7]	46.21	3.2	3150	450
DARTS [28]	16.32	3.3	1.5	1
P-DARTS [60]	45.24	5.3	0.3	1
PC-DARTS [40]	45.53	6.2	0.1	1
GDAS [27]	42.21	2.5	0.3	1
EX-Net(CIFAR-10)(ours)	44.56	2.0	0.02	1
EX-Net(CIFAR-100)(ours)	46.87	4.3	0.02	1

shows competitive test accuracy while exhibiting an advantage in terms of the number of parameters. Furthermore, EX-Net greatly reduces the search time cost and the required computational resource consumption.

3) Comparison With Automatic NAS Methods

Compared to the competing completely automatic NAS algorithms, EX-Net exhibits great superiority over Large-scale Evolution and NAS in terms of the accuracy and number of parameters. In addition, EX-Net consumes only 0.02 GPU Days, which is substantially less than Large-scale Evolution and NAS. Additionally, the GPU resources required by EX-Net are 800x less than those required by NAS. EX-Net has better test accuracy and fewer parameters than AmoebaNet-A. The GPU Days required by EX-Net is only 0.02, which is 1/157,500 of that needed by AmoebaNet-A, and the computational resources required by the GPU are just 1/450 of those demanded by AmoebaNet-A. EX-Net is better than AE-CNN in terms of the mean and deviation of test accuracy and the number of parameters on CIFAR-10 and CIFAR-100. EX-Net can obtain a better improvement in the search time cost and the required GPU resource consumption. Compared to CNN-GA, EX-Net has a higher test accuracy on CIFAR-10 and has fewer parameters. In addition, EX-Net achieves better accuracy on the more complex CIFAR-100 dataset, while the number of parameters is close to that of CNN-GA. The search time of EX-Net

is approximately 1/1750 of that consumed by CNN-GA. NSGA-Net attains slightly better accuracy than EX-Net on CIFAR-10 (97.5% vs. 96.95%), but EX-Net has smaller deviations of test accuracy (0.07 vs. 0.10), and only consumes 1/13 of the parameters required by NSGA-Net (2.0M vs. 26.8M). When using the same computational resources, the search time of EX-Net is 200x less than that of NSGA-Net. Compared to LF-MOGP, EX-Net can achieve a significant advantage in terms of accuracy on both datasets. Moreover, EX-Net only consumes 0.02 GPU Days, which is 500x less than LF-MOGP. Therefore, EX-Net shows significant advantages over automatic algorithms in all objectives.

4) Discussion

In summary, EX-Net outperforms most manually-designed network architectures in terms of the test accuracy while requiring fewer parameters. EX-Net also shows excellent advantages over most of the automatic NAS algorithms regarding test accuracy and the number of parameters. The proposed approach also requires fewer GPU resources and achieves 200x to 1,120,000x search time reductions. Compared to the semi-automatic NAS algorithms, the test accuracy advantage of EX-Net is not apparent since other algorithms involve manual fine-tuning. However, EX-Net can achieve more minor deviations in test accuracy, which demonstrates the robustness and stability of the EX-Net. Furthermore, EX-Net utilizes fewer parameters, and the search time cost and computational resource consumption

are significantly reduced, which is the primary purpose of the work in this paper.

Table II shows that each run by LoNAS has a similar search time on CIFAR-10 and CIFAR-100, demonstrating the robustness of the proposed algorithm regarding the time cost. By comprehensively considering the test accuracies and the number of parameters, Tables III and IV show the best network architectures discovered by the proposed algorithm on the CIFAR-10 and CIFAR-100 datasets. They also show that the best network architecture obtained on CIFAR-10 is composed of three Reg Units with 34 convolutional layers. The network FLOPs are 0.91 G. The best network architecture obtained on CIFAR-100 comprises four Reg Units of 46 convolutional layers, and the FLOPs of the network are 1.05 G.

E. Transferability

The ImageNet-16-120 dataset [65] is also considered for investigating the transferability of the network architectures searched by LoNAS on the CIFAR-10 and CIFAR-100 datasets. The same training settings as mentioned in Section IV-A are used. The comparison results in Table V show that the network architecture searched on CIFAR-10 obtains better test accuracy than most manually-designed and automatic architectures. The network architecture searched on CIFAR-100 achieves the best test accuracy on ImageNet-16-120 while consuming minimal GPU Days and GPU resources. The results indicate that LoNAS has good transferability to other datasets.

V. Conclusion

This paper proposes a LoNAS method that can quickly search for network architectures with high accuracy and few parameters. The computational resources required by LoNAS to find the network architectures are also low.

In LoNAS, a Reg Block is proposed based on group convolution and the SENet module, which helps improve the accuracy while reducing the number of parameters. A variable-architecture

encoding strategy based on the Reg Block is designed to construct an expanded search space. A training-free proxy is proposed to evaluate individuals based on the NTK, reducing the time and computational resource costs. Furthermore, a three-stage EA based on multiple-criteria environmental selection is designed to balance the exploration and exploitation of the proposed algorithm. A set of mutation operators is proposed and applied to the Reg Block to explore more of the search space.

The proposed algorithm is examined on two representative benchmark datasets, CIFAR-10 and CIFAR-100, and compared with various state-of-the-art algorithms, including manually-designed network architectures, semiautomatic network architectures, and automatic network architectures. The experimental results show that the network architectures found by LoNAS outperform most manually-designed and automatic architectures in terms of classification performance and the number of parameters. The network architectures also achieve the competitive performance on the semiautomatic architectures in terms of test accuracy while improving most algorithms in the number of required parameters. More importantly, LoNAS shows significant advantages in reducing the search time cost and computational resource consumption. Finally, the architectures found on CIFAR-10 and CIFAR-100 can be transferred to ImageNet-16-120 with good performance.

In LoNAS, two computational constraints are designed to limit the number of network architecture parameters in the search space; these constraints compress the search space and reduce the diversity to some extent. Future work will explore other methods to balance the trade-off between multiple objectives in larger search spaces. In addition, the NTK should be improved since the performance of the NTK conditions fluctuates.

Acknowledgment

This work was supported in part by the National Natural Science Foundation of China under Grants 62073155, 62002137, 62106088, and 62206113; in part by Jiangsu University, China

through Blue Project; and in part by Guangdong Provincial Key Laboratory under Grant 2020B121201001.

References

- [1] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *Proc. Int. Conf. Learn. Representations*, 2015.
- [2] 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.
- [3] C. Szegedy et al., "Going deeper with convolutions," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2015, pp. 1–9.
- [4] 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.
- [5] B. Baker, O. Gupta, N. Naik, and R. Raskar, "Designing neural network architectures using reinforcement learning," 2016, *arXiv:1611.02167*.
- [6] A. Brock, T. Lim, J. Ritchie, and N. Weston, "SMASH: One-shot model architecture search through hypernetworks," in *Proc. Int. Conf. Learn. Representations*, 2018.
- [7] E. Real, A. Aggarwal, Y. Huang, and Q. V. Le, "Regularized evolution for image classifier architecture search," in *Proc. AAAI Conf. Artif. Intell.*, 2019, vol. 33, pp. 4780–4789.
- [8] Y. Sun, B. Xue, M. Zhang, and G. Yen, "Completely automated CNN architecture design based on blocks," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 31, no. 4, pp. 1242–1254, Apr. 2020.
- [9] Y. Sun, B. Xue, M. Zhang, G. G. Yen, and J. Lv, "Automatically designing CNN architectures using the genetic algorithm for image classification," *IEEE Trans. Cybern.*, vol. 50, no. 9, pp. 3840–3854, Sep. 2020.
- [10] Z. Sun, M. Lin, X. Sun, Z. Tan, H. Li, and R. Jin, "MAE-Det: Revisiting maximum entropy principle in zero-shot NAS for efficient object detection," in *Proc. Int. Conf. Mach. Learn.*, 2022, pp. 20810–20826.
- [11] S. Liu et al., "EVSrNet: Efficient video super-resolution with neural architecture search," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2021, pp. 2480–2485.
- [12] S. Xu and H. Quan, "ECT-NAS: Searching efficient CNN-transformers architecture for medical image segmentation," in *Proc. IEEE Int. Conf. Bioinf. Biomed.*, 2021, pp. 1601–1604.
- [13] 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.
- [14] E. Real et al., "Large-scale evolution of image classifiers," in *Proc. Int. Conf. Mach. Learn.*, 2017, pp. 2902–2911.
- [15] A. G. Howard et al., "MobileNets: Efficient convolutional neural networks for mobile vision applications," 2017, *arXiv:1704.04861*.
- [16] 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.
- [17] Z. Lu et al., "NSGA-Net: Neural architecture search using multi-objective genetic algorithm," in *Proc. Genet. Evol. Comput. Conf.*, 2019, pp. 419–427.
- [18] T. Elsen, J. H. Metzen, and F. Hutter, "Efficient multi-objective neural architecture search via Lamarckian evolution," in *Proc. Int. Conf. Learn. Representations*, 2018.
- [19] T. Back, *Evolutionary Algorithms in Theory and Practice: Evolution Strategies, Evolutionary Programming, Genetic Algorithms*. London, U.K.: Oxford Univ. Press, 1996.
- [20] L. Xie and A. Yuille, "Genetic CNN," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2017, pp. 1379–1388.

- [21] O. Russakovsky et al., "ImageNet large scale visual recognition challenge," *Int. J. Comput. Vis.*, vol. 115, no. 3, pp. 211–252, 2015.
- [22] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2012, vol. 25, pp. 1097–1105.
- [23] T. Zhang, C. Lei, Z. Zhang, X.-B. Meng, and C. P. Chen, "AS-NAS: Adaptive scalable neural architecture search with reinforced evolutionary algorithm for deep learning," *IEEE Trans. Evol. Comput.*, vol. 25, no. 5, pp. 830–841, Oct. 2021.
- [24] H. Cai, L. Zhu, and S. Han, "ProxylessNAS: Direct neural architecture search on target task and hardware," in *Proc. Int. Conf. Learn. Representations*, 2018.
- [25] B. Wu et al., "FBNet: Hardware-aware efficient convnet design via differentiable neural architecture search," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 10734–10742.
- [26] Z. Lu, K. Deb, E. Goodman, W. Banzhaf, and V. N. Boddeh, "NSGANetV2: Evolutionary multi-objective surrogate-assisted neural architecture search," in *Proc. Eur. Conf. Comput. Vis.*, 2020, pp. 35–51.
- [27] X. Dong and Y. Yang, "Searching for a robust neural architecture in four GPU hours," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 1761–1770.
- [28] H. Liu, K. Simonyan, and Y. Yang, "Darts: Differentiable architecture search," in *Proc. Int. Conf. Learn. Representations*, 2018.
- [29] C. Liu et al., "Auto-deeplab: Hierarchical neural architecture search for semantic image segmentation," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 82–92.
- [30] 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.
- [31] K. Yu, C. Sciuto, M. Jaggi, C. Musat, and M. Salzmann, "Evaluating the search phase of neural architecture search," in *Proc. Int. Conf. Learn. Representations*, 2020.
- [32] S. Xie, A. Kirillov, R. Girshick, and K. He, "Exploring randomly wired neural networks for image recognition," in *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, 2019, pp. 1284–1293.
- [33] C. Liu et al., "Progressive neural architecture search," in *Proc. Eur. Conf. Comput. Vis.*, 2018, pp. 19–34.
- [34] H. Cai, C. Gan, T. Wang, Z. Zhang, and S. Han, "Once-for-all: Train one network and specialize it for efficient deployment," in *Proc. Int. Conf. Learn. Representations*, 2019.
- [35] R. Luo, X. Tan, R. Wang, T. Qin, E. Chen, and T.-Y. Liu, "Semi-supervised neural architecture search," in *Proc. Adv. Neural Inf. Process. Syst.*, 2020, vol. 33, pp. 10547–10557.
- [36] J. Mellor, J. Turner, A. Storkey, and E. J. Crowley, "Neural architecture search without training," in *Proc. Int. Conf. Mach. Learn.*, 2021, pp. 7588–7598.
- [37] W. Chen, X. Gong, and Z. Wang, "Neural architecture search on imagenet in four GPU hours: A theoretically inspired perspective," in *Proc. Int. Conf. Learn. Representations*, 2020.
- [38] A. Jacot, F. Gabriel, and C. Hongler, "Neural tangent kernel: Convergence and generalization in neural networks," in *Proc. 32nd Int. Conf. Neural Inf. Process. Syst.*, 2018, pp. 8580–8589.
- [39] B. Hanin and M. Nica, "Finite depth and width corrections to the neural tangent kernel," in *Proc. Int. Conf. Learn. Representations*, 2019.
- [40] Y. Xu et al., "PC-darts: Partial channel connections for memory-efficient architecture search," in *Proc. Int. Conf. Learn. Representations*, 2019.
- [41] S.-Y. Huang and W.-T. Chu, "Searching by generating: Flexible and efficient one-shot NAS with architecture generator," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2021, pp. 983–992.
- [42] A. Klos, M. Rosenbaum, and W. Schifffmann, "Neural architecture search based on genetic algorithm and deployed in a bare-metal kubernetes cluster," *Int. J. Netw. Comput.*, vol. 12, no. 1, pp. 164–187, 2022.

- [43] J. K. Kim, W. Ahn, S. Park, S.-H. Lee, and L. Kim, "Early prediction of sepsis onset using neural architecture search based on genetic algorithms," *Int. J. Environ. Res. Public Health*, vol. 19, no. 4, 2022, Art. no. 2349.
- [44] M. Lin et al., "ZEN-NAS: A zero-shot NAS for high-performance image recognition," in *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, 2021, pp. 347–356.
- [45] Y. Sun, B. Xue, M. Zhang, and G. G. Yen, "Evolving deep convolutional neural networks for image classification," *IEEE Trans. Evol. Comput.*, vol. 24, no. 2, pp. 394–407, Apr. 2020.
- [46] L. Zhao and W. Fang, "An efficient and flexible automatic search algorithm for convolution network architectures," in *Proc. IEEE Congr. Evol. Comput.*, 2021, pp. 2203–2210.
- [47] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *Proc. Int. Conf. Mach. Learn.*, 2015, pp. 448–456.
- [48] J. Hu, L. Shen, and G. Sun, "Squeeze-and-excitation networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 7132–7141.
- [49] X. Glorot, A. Bordes, and Y. Bengio, "Deep sparse rectifier neural networks," in *Proc. 14th Int. Conf. Artif. Intell. Statist. Workshop*, 2011, pp. 315–323.
- [50] C. Cao et al., "Look and think twice: Capturing top-down visual attention with feedback convolutional neural networks," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2015, pp. 2956–2964.
- [51] M. Jaderberg et al., "Spatial transformer networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2015, vol. 28, pp. 2017–2025.
- [52] Y. Shin and G. E. Karniadakis, "Trainability of ReLU networks and data-dependent initialization," *J. Mach. Learn. Model. Comput.*, vol. 1, no. 1, pp. 39–74, 2020.
- [53] R. Mallipeddi, P. N. Suganthan, Q.-K. Pan, and M. F. Tasgetiren, "Differential evolution algorithm with ensemble of parameters and mutation strategies," *Appl. Soft Comput.*, vol. 11, no. 2, pp. 1679–1696, 2011.
- [54] X.-G. Zhou, G.-J. Zhang, X.-H. Hao, L. Yu, and D.-W. Xu, "Differential evolution with multi-stage strategies for global optimization," in *Proc. IEEE Congr. Evol. Comput.*, 2016, pp. 2550–2557.
- [55] T. DeVries and G. W. Taylor, "Improved regularization of convolutional neural networks with cutout," 2017, *arXiv:1708.04552*.
- [56] G. Larsson, M. Maire, and G. Shakhnarovich, "Fractalnet: Ultra-deep neural networks without residuals," 2016, *arXiv:1605.07648*.
- [57] S. Zagoruyko and N. Komodakis, "Wide residual networks," in *Proc. Brit. Mach. Vis. Conf., Assoc.*, 2016, pp. 87.1–87.12.
- [58] S. Xie, R. Girshick, P. Dollár, Z. Tu, and K. He, "Aggregated residual transformations for deep neural networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 1492–1500.
- [59] H. Liu, K. Simonyan, O. Vinyals, C. Fernando, and K. Kavukcuoglu, "Hierarchical representations for efficient architecture search," in *Proc. Int. Conf. Learn. Representations*, 2018.
- [60] X. Chen, L. Xie, J. Wu, and Q. Tian, "Progressive differentiable architecture search: Bridging the depth gap between search and evaluation," in *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, 2019, pp. 1294–1303.
- [61] Z. Zhao, Y. Kang, A. Hou, and D. Gan, "SP-DARTS: Synchronous progressive differentiable neural architecture search for image classification," *IEICE Trans. Inf. Syst.*, vol. 104, no. 8, pp. 1232–1238, 2021.
- [62] Z. Zhong, J. Yan, W. Wu, J. Shao, and C.-L. Liu, "Practical block-wise neural network architecture generation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 2423–2432.
- [63] B. Zoph and Q. Le, "Neural architecture search with reinforcement learning," in *Proc. Int. Conf. Learn. Representations*, 2017.
- [64] Q. Liu, X. Wang, Y. Wang, and X. Song, "Evolutionary convolutional neural network for image classification based on multi-objective genetic programming with leader-follower mechanism," *Complex Intell. Syst.*, pp. 1–18, 2022.
- [65] P. Chrabaszcz, I. Loshchilov, and F. Hutter, "A downsampled variant of imagenet as an alternative to the CIFAR datasets," 2017, *arXiv:1707.08819*.




Share Your Preprint Research with the World!

TechRxiv is a free preprint server for unpublished research in electrical engineering, computer science, and related technology. TechRxiv provides researchers the opportunity to share early results of their work ahead of formal peer review and publication.

BENEFITS:

- Rapidly disseminate your research findings
- Gather feedback from fellow researchers
- Find potential collaborators in the scientific community
- Establish the precedence of a discovery
- Document research results in advance of publication

Upload your unpublished research today!

 Follow us @TechRxiv_org
Learn more techrxiv.org

TechRxiv™
Powered by IEEE