Enhanced Gradient for Differentiable Architecture Search

Haichao Zhang, Kuangrong Hao, Lei Gao, *Member, IEEE*, Xue-song Tang, and Bing Wei, *Member, IEEE*

Abstract—**In recent years, neural architecture search (NAS) methods have been proposed for the automatic generation of task-oriented network architecture in image classification. However, the architectures obtained by existing NAS approaches are optimized only for classification performance and do not adapt to devices with limited computational resources. To address this challenge, we propose a neural network architecture search algorithm aiming to simultaneously improve the network performance and reduce the network complexity. The proposed framework automatically builds the network architecture at two stages: block-level search and network-level search. At the stage of block-level search, a gradient-based relaxation method is proposed, using an enhanced gradient to design high-performance and low-complexity blocks. At the stage of network-level search, an evolutionary multiobjective algorithm is utilized to complete the automatic design from blocks to the target network. The experimental results demonstrate that our method outperforms all evaluated hand-crafted networks in image classification, with an error rate of 3.18% on Canadian Institute for Advanced Research (CIFAR10) and an error rate of 19.16% on CIFAR100, both at network parameter size less than 1 M. Obviously, compared with other NAS methods, our method offers a tremendous reduction in designed network architecture parameters.**

Index Terms—**Convolutional neural networks (CNNs), evolutionary deep learning, multiobjective evolutionary optimization, neural architecture search (NAS).**

I. Introduction

OVER the past decades, convolutional neural network (CNN) has shown remarkable success in almost all aspects of computer vision. It is first successfully applied in image classification, where many CNN architectures are developed, including AlexNet [1], GoogLeNet [2], ResNet [3], and DenseNet [4]. Meanwhile, architectures such as ShuffleNet [5] and MobileNet [6] are designed to enable the actual deployment of a high-performance model on resource-constrained devices. In most cases, CNN needs to be designed based on a specific task, which leads to endless hyperparameter tuning. To alleviate the tedious work, the neural architecture search (NAS) [7], [8], [9] is proposed and it is proven as a promising approach to pose the process of designing neural network architecture as an optimization problem [10]. This automatic and efficient approach for designing neural network architecture has a significant impact on the implementation of neural networks. For example, the design algorithm of neural network architecture improves the performance of CNN significantly, enabling CNN [9], [11], [12], [13] to achieve outstanding performance on multiple datasets such as ImageNet [14] and ChexRay [15]. In addition, the CNN architecture design algorithm also promotes the wide use of CNN in different fields [16], [17]. This is because users who are familiar with data often have no experience in designing CNN architectures, but they can also design CNNs for specific tasks through architecture design algorithms. Therefore, the CNN automation architecture is widely interested and concerned by users who have limited CNN domain knowledge [18].

So far, there are mainly three pathways for NAS. One popular way to search for network architectures is utilizing evolutionary algorithms [19]. This method generates new individuals through evolutionary operations (e.g., crossover and mutation) from the search space and obtains the best-performing individuals through population iteration in the search space. Such as Sun et al. [9] implemented the automatic evolving CNN (AE-CNN) which uses a genetic algorithm (GA) based on ResNet [3] and DenseNet [4] blocks to automatically evolve CNN. An end-to-end random forest-based predictor is proposed to optimize neural network architecture by accelerating the fitness evaluation in evolutionary deep learning and showed good performance in terms of classification accuracy and computing resources [20]. Lu et al. [11] proposed a multiobjective evolutionary algorithm for NAS based on NSGA-II. This method approximates the entire Pareto boundary by the genetic operations of recombining and modifying architecture components gradually. The reinforcement learning method [21] is another popular approach, which applies a recurrent network as the controller to generate the network architecture and optimize the controller through reinforcement learning [12]. Although the performance of the above two kinds of methods is impressive, the search process is significantly resource-intensive for a large dataset like ImageNet [14]. Therefore, the method based on the gradient search

Manuscript received 20 March 2021; revised 2 December 2021 and 11 October 2022; accepted 30 December 2022. This work was supported in part by the Fundamental Research Funds for the Central Universities under Grant 2232021A-10 and Grant 2232021D-37, in part by the National Natural Science Foundation of China under Grant 61903078, in part by the Natural Science Foundation of Shanghai under Grant 20ZR1400400 and Grant 21ZR1401700, and in part by the Fundamental Research Funds for the Central Universities and Graduate Student Innovation Fund of Donghua University under Grant CUSF-DH-D-2021048. *(Corresponding author: Kuangrong Hao.)*

Haichao Zhang, Kuangrong Hao, Xue-song Tang, and Bing Wei are with the Engineering Research Center of Digitized Textile and Apparel Technology, Ministry of Education, and the College of Information Science and Technology, Donghua University, Shanghai 201620, China (e-mail: krhao@dhu.edu.cn).

Lei Gao is with the Commonwealth Scientific and Industrial Research Organization (CSIRO), Waite Campus, Urrbrae, SA 5064, Australia.

This article has supplementary material provided by the authors and color versions of one or more figures available at https://doi.org/10.1109/TNNLS.2023.3235479.

Digital Object Identifier 10.1109/TNNLS.2023.3235479

network architecture has attracted wide attention because it greatly accelerates the search process and achieves impressive performance [22]. The third method, differentiable architecture search (DARTS) [23], is a recently proposed approach with outstanding performance. By relaxing the discrete search space to be continuous, DARTS formulates the process of searching optimal neural network architecture as a bilevel optimization method. It obtains the optimized architecture with respect to its performance on the validation set by gradient descent. Although DARTS has achieved outstanding performance with great search acceleration, it still has two deficiencies. DARTS uses a proxy network composed of fewer blocks to search the structure of normal blocks and reduction blocks, and then, the obtained block structure is applied/shared in all blocks of the target network. In this case, the complexity of the network is not considered in the process of searching for block structure in the proxy network, which leads to the redundancy of block structure complexity. Furthermore, the shared blocks are not optimally designed in the target network, which causes the architecture redundancy of the target network.

In this work, we adopt a new viewpoint to address the aforementioned deficiencies. We propose a multiobjective NAS algorithm—enhanced gradient for differentiable architecture search (EG-DARTS). The proposed approach takes into account both network performance (classification accuracy in this work) and network complexity when calculating the gradient during searching the architecture. Furthermore, to optimize the architecture of the shared block in the target network, we design a multiobjective evolutionary optimization method based on nondominated sorting to construct the target network. This evolutionary algorithm of nondominated sorting is built on the standard GA, with the selection regeneration method improved in the following way: each individual is stratified according to its dominated and nondominated relationships, and then, the selection operation is done, which makes the algorithm obtain satisfactory results in multiobjective evolutionary optimization. Regarding the dominant and nondominant relationship, readers can refer to [24]. Therefore, our algorithm fully considers the adaptation of the blocks searched in the proxy network to the target network. The designed algorithm is an end-to-end approach to construct the neural networks, which fully considers the high performance of the network while consuming limited computing resources. This algorithm helps CNN to achieve high-performance actual deployment on resource-constrained devices. The key contributions made in this article are summarized as follows.

1) Based on the continuously relaxed search space [25], a DARTS method is established for dual objectives to construct low-complexity, high-performance CNN blocks.
2) An evolutionary optimization method based on the nondominated ranking is introduced to construct low-complexity, high-performance target networks on CNN shared blocks. The proposed method reduces the gap between the proxy network and the target network and helps designers to build target networks for feasible deployment in devices with limited computational resources.
3) The proposed method provides excellent network search performance and is tested on datasets Canadian Institute for Advanced Research (CIFAR10) and CIFAR100. Additionally, we analyze the relationship among network parameters, floating-point operations (FLOPS), and interface time. The results show that the inference time is directly related to the network depth.

The rest of this article is organized as follows. Section II summarizes relevant works in the literature. In Section III, we provide a detailed description of the main components of our approach. We describe the experimental setup to validate our approach along with a discussion of the results in Section IV. Finally, we conclude with a summary of our findings and present possible future directions in Section V.

II. Literature Review

Since the twentieth century, research has been devoted to the automation of neural network design. GAs [26] or other evolutionary algorithms have been applied to search for high-performance architectures, such as the works that evolved the topology and hyperparameters of neural networks. In recent years, there has been an increasing interest in searching neural network architecture, and the methods based on evolutionary computation have continuously shown satisfying performance in doing so [27]. Similarly, reinforcement learning-based NAS has also demonstrated excellent performance, such as the pioneering work that adopted the recurrent neural network (RNN) as the controller to sequentially design the architecture [28]. The gradient-based NAS [23], [25] method utilizes the gradient of the network architecture parameters to optimize the architecture.

This method greatly accelerates the search process to obtain a network architecture with favorable performance. The above streams are the main search strategies frequently used in NAS. Next, we review the studies of neural network architecture search for image classification and present some operations in NAS that are utilized as the foundation of our proposed approach.

A. NAS Based on Blocks

Deep neural networks have complex discrete architecture parameters and continuous hyperparameters, so optimizing the architecture of deep neural networks is an arduous task. Many studies [17] have simplified the process of NAS into searching the shared structural blocks of networks. Originally, Zoph et al. [29] proposed NASNet, which searches for architectural building blocks on a smaller dataset and then transfers the blocks to a larger dataset. The portability of the blocks proposed by NASNet provides new ideas for NAS. Sun et al. [9] used the GA to automatically evolve neural network architectures based on the blocks.

The block-based optimization has also been applied to reinforcement learning-based NAS. Zhong et al. [12] proposed BlockQNN, which provides the blockwise network to generate blocks that are stacked to construct the whole target network. BlockQNN utilizes the Q-learning paradigm with the epsilon-greedy exploration approach to build the shared blocks and accelerate the NAS process. Additionally, limiting the search space, early-stop strategy [30], progressive searching [31], and weight sharing in architecture were proposed to speed up the search process. Unfortunately, the NAS methods [17] based on reinforcement learning are still resource hungry and suffer from limited search space because of the fixed-length coding of network structure.

Most gradient-based NAS algorithms optimize the neural network architecture through the block-based approach. Liu et al. [23] proposed the DARTS algorithm, which optimizes the target network by optimizing the blocks in the proxy network. As an efficient framework of NAS, DARTS has been widely used in polarimetric synthetic aperture radar (PolSAR) image classification [32]. However, searching in the proxy network is low level due to the disparity between the proxy and the target network, and the shared blocks are not optimally designed in the target network. Direct sparse optimization NAS (DSO-NAS) [33] regards the optimization of the neural network architecture as the view of model pruning. DOS-NAS utilizes a scaling factor to scale the information flow between operations to optimize an initial fully connected block, and then, sparse regularization is applied to pure useless connections in the blocks. However, the optimization of network architecture complexity is not involved in DSO-NAS.

B. Multiobjective for NAS

Multiobjective optimization is concerned with simultaneous optimization where tradeoffs exist between two or more conflicting objectives [34], [35]. Neuro-evolution with multiobjective optimization (NEMO), proposed by Kim et al. [36], is one of the earliest multiobjective evolutionary methods for evolving CNN architecture. NEMO applies NSGA-II [24] to maximize network classification accuracy and minimize the inference time. In the NEMO method, the number of output channels from each layer is searched within a restricted space of seven different structures. Based on this work [31], Dong et al. [37] proposed device-aware progressive search for pareto-optimal neural architectures, optimizing for both device-related (DPP-Net), which takes a compact architecture search space driven by mobile CNN blocks and further improves search efficiency by adopting resource consumption. Lamarckian evolutionary algorithm for multi-objective neural architecture dEsign (LEMONADE) reduces computational requirements through a proxy network and allows the newly generated architecture to share parameters with its forerunners, avoiding the consumption of computing resources caused by training the new network from scratch. However, the LEMONADE on the CIFAR dataset requires almost 100 days of computing time. Lu et al. [11] proposed a multiobjective NAS method based on NSGA-II, which approximates the entire Pareto boundary by reorganizing and modifying the architecture of the network. Meanwhile, Huang et al. [38] proposed multi-objective evolutionary compression learning (MOE-CL), which took the network classification error rate and compression rate as the two objectives of NAS, and proposed a mechanism to design an approximate compressed model generation, which reduces the high network training cost involved in the optimization process. Correspondingly, Jiang et al. [39] proposed a decomposition-based multiobjective particle swarm optimization (MOPSO/D-net) approach for NAS. The method combines hybrid binary coding and adaptive penalty-based bounded intersection MOPSO/D to solve the formulated multiobjective NAS.

III. Proposed Method

In this section, the details of the proposed framework are presented. We first give an overview of the proposed EG-DARTS. Then, the design of the search space is given. Next, the core differentiable gradient-enhanced architecture search approach to construct the blocks is described. Finally, we combine the blocks with the multiobjective evolutionary architecture search algorithm to build the target network.

A. Algorithm Overview

The actual application of NAS not only considers the objective of maximizing performance but also evaluates it against the conflicting objective in deployment scenarios within constrained computing resources. Therefore, our proposed algorithm will design high-performance neural network architectures with diverse computational complexity for resource-constrained scenarios. We regard the task as a multiobjective bilevel optimization problem, which can be expressed mathematically as

$$\begin{aligned} &\min\ F(\alpha) = (\mathcal{L}_{\text{val}}(\omega^*(\alpha), \alpha), \mathcal{C}(\alpha))^{\mathrm{T}} \\ &\text{s.t.}\quad \omega^*(\alpha) = \arg\min_{\omega\epsilon\Omega} \mathcal{L}_{\text{train}}(\omega, \alpha) \end{aligned} \tag{1}$$

where α is the network architecture parameters and ω are the weights of the network. The lower level objective $\mathcal{L}_{\text{train}}(\omega, \alpha)$ is the cross-entropy loss function on the training data. $\omega^*(\alpha)$ is the weights of the network where $\mathcal{L}_{\text{train}}(\omega, \alpha)$ is maximized.

The upper level objective $F(\alpha)$ consists of the loss function $\mathcal{L}_{\text{val}}(\omega^*(\alpha), \alpha)$ on the validation data and the computational complexity $\mathcal{C}(\alpha)$ of the network architecture. To describe the complexity of the network architecture more accurately, the parameter size of the network is utilized to represent the complexity of the network. The goal for the multiobjective bilevel architecture optimization is to find $\alpha^{\#}$ that minimizes the validation loss $\mathcal{L}_{\text{val}}(\omega^*(\alpha), \alpha)$ and the parameter size of the network architecture.

The purpose of the proposed EG-DARTS is to design high-performance architecture with diverse complexities for different specific scenarios. EG-DARTS consists of two stages and its schematic is shown in Fig. 1. EG-DARTS starts with the stage that we call the "block-level search stage," which is to search the optimal blocks from the search space. At this stage, as shown in Fig. 1(a)–(c), the block is designed by the gradient-based method, which is based on the progressive DARTS (PDARTS) approach [28] and cooperates with architecture compression. As shown in Fig. 1(c), the reduction block and the normal block are searched in the proxy network. Likewise, another stage is called the "network-level search stage," which is to optimize the target network composed of obtained blocks. As shown in Fig. 1(e) and (f), the multiobjective evolutionary optimization method is combined with the objective of high performance and low complexity, and the target network is constructed based on the blocks obtained from the proxy network. Finally, the Pareto set $P_q = \{\alpha_1^q, \alpha_2^q, \ldots, \alpha_i^q, \ldots, \alpha_n^q\}$ with diverse computational

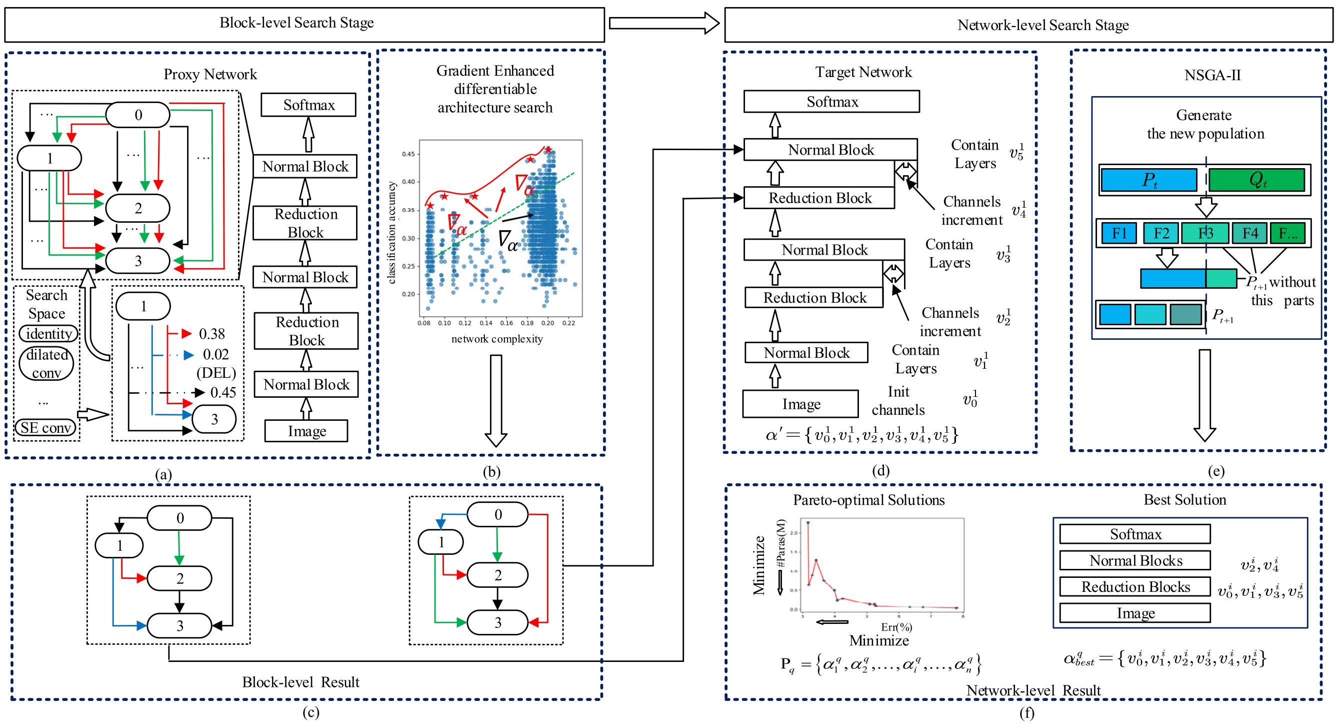

Fig. 1. Schematic of the EG-DARTS search process: (a) proxy network is composed of convolution operations and stacked blocks; (b) method of gradient-enhanced DARTS to maximize network classification accuracy and minimize network complexity; (c) normal block and reduction block obtained from the block-level search; (d) target network is constructed based on block parameterization; (e) multiobjective evolutionary optimization approach; and (f) Pareto set P_q with diverse computational complexity is constructed by the proposed approach.

complexity is constructed using NSGA-II as Fig. 1(f). In the remainder of this section (Sections III-B–III-D), a detailed description of the aforementioned components will be provided.

B. Search Space and Encoding

NAS is to design the optimal network architecture from different search spaces. The versatility of the search space has a major impact on the quality of the optimal architecture. To better design the search space, we need to understand the basic composition of neural networks. In the modern classic CNN, its architecture is mostly composed of block-level design and network-level design. Block-level design refers to hierarchical connection methods and calculation operations. Correspondingly, network-level design refers to changes in width, depth, and spatial resolution. Many manually designed networks are based on this idea, such as ResNet [3] and Inception [2]. Similarly, various NAS methods are also searching for the best blocks. A block is a small module in CNN that is repeated multiple times to form the entire neural network. Therefore, choosing the search space should be oriented to block-level design. In this work, we leverage the search space of P-DARTS [25] as the baseline framework. In order to build a scalable network architecture, there are two types of blocks we build: 1) normal block, which returns feature maps with the same spatial resolution and 2) reduction block, which returns feature maps with spatial resolution halved by a stride of two.

We search structure on the proxy network with fewer building blocks. The goal of designing the search space is to obtain blocks with low computational complexity and high performance. The block is defined as a directed acyclic graph (DAG) of N nodes, and each node x_i is a candidate representative of feature mapping in the network. The search space is denoted as $\mathcal{O}$, where each operation represents a candidate function $o(\cdot)$. Similarly, each directed edge (i, j) is the candidate operation $o(x_i)$. Likewise, the block can be assumed to obtain a two-input node and a single-output node. The input nodes are defined as block outputs in the previous two blocks. Therefore, an edge (i, j) consists of a series of operations weighted $\alpha^{(i,j)}$ by architecture parameters, and it can be formulated as

$$f_{i,j}(x_i) = \sum_{o\epsilon\mathcal{O}_{(j,j)}} \frac{\exp(\alpha_o^{(i,j)})}{\sum_{o'\epsilon\mathcal{O}} \exp(\alpha_{o'}^{(i,j)})} o(x_i) \tag{2}$$

where $i < j$, and the intermediate node is $x_j = \sum_{i<j} F_{i,j}(x_i)$. Thus, the output node is $x_{N-1} = \text{concat}(x_2, x_3, \ldots, x_{N-2})$, where concat$(\cdot)$ represents that all input signals in the channel dimension are concatenated. For the candidate convolution operations, we choose the computational operations between nodes from the following options, which are collected based on the excellent performance in the study of CNNs:

1) identity (skip-connect);
2) 3×3 max pooling (max_pool_3×3);
3) 3×3 average pooling (avg_pool_3×3);
4) 3×3 efficient channel attention (EcaNet_3×3);
5) 3×3 depthwise-separable convolution (sep_conv_3×3);
6) 5×5 depthwise-separable convolution (sep_conv_5×5);
7) 7×7 depthwise-separable convolution (sep_conv_7×7);
8) 3×3 dilated convolution (dil_conv_3×3);
9) 5×5 dilated convolution (dil_conv_5×5);
10) 3×3 local binary convolution (Lbcnn_3×3);
11) 5×5 local binary convolution (Lbcnn_5×5);

12) 1×7 then 1×7 convolution (conv_7 × 1_1 × 7).

As shown in Fig. 1(a), during the algorithm initialization at the first stage, the connection between two nodes is the operations of the search space described above, and there are 12 operations between the two nodes. By the method of gradient-enhanced DARTS, the result block contains only one computational operation per two nodes.

C. Blocks-Level Search Strategy

The purpose of the search at the first stage is to obtain the normal blocks and reduction blocks. The task of architecture search in the proposed gradient-based method can be understood as designing blocks with high performance and low complexity from the search space $\mathcal{O}$, through learning a set of continuous variables $\alpha^* = \{\alpha^{(i,j)}\}$. According to (1), the discrete architecture can be obtained by pruning the mixed computational operations, i.e., $o^{(i,j)} = \arg\max_{o \epsilon \mathcal{O}} \alpha_o^{(i,j)}$. It is difficult to optimize network architecture performance and complexity based on gradients. To simplify the process, we first minimize validation loss from (1) and it can be simplified as

$$\nabla_\alpha \mathcal{L}_{\text{val}} \left(\omega^*(\alpha), \alpha\right) \approx \nabla_\alpha \mathcal{L}_{\text{val}} \left(\omega - \varepsilon \nabla_\omega \mathcal{L}_{\text{train}} \left(\omega, \alpha\right), \alpha\right) \quad (3)$$

where ω represents the weights of the current network and ϵ represents the learning rate at each step of the architecture optimization process. Then, (3) can be expended by the chain rule as

$$\nabla_\alpha \mathcal{L}_{\text{val}} \left(\omega', \alpha\right) - \varepsilon \frac{\nabla_\alpha \mathcal{L}_{\text{train}} \left(\omega^+, \alpha\right) - \nabla_\alpha \mathcal{L}_{\text{train}} \left(\omega^-, \alpha\right)}{2\epsilon} \quad (4)$$

where $\epsilon = 0.01/|\nabla_{\omega'} \mathcal{L}_{\text{val}}(\omega', \alpha)$ represents a small scalar and $\omega^\pm = \omega \pm \epsilon \nabla_{\omega'} \mathcal{L}_{\text{val}}(\omega', \alpha)$. $w' = w - \xi \nabla_w \mathcal{L}_{\text{train}}(w, \alpha)$ is the weight of the one-step forward network. Since (4) simplifies the architecture search algorithm, the complexity of NAS is reduced from to $O(|\alpha||\omega|)$ to $O(|\alpha| + |\omega|)$.

To minimize the classification loss and minimize the complexity of the network architecture, we propose an improvement to the gradient ∇_α updating of the architecture parameters $\alpha^{(i,j)}$ as shown in Fig. 1(b). The relationship between classification loss and the network complexity is assumed to be approximated as

$$y = h_\theta(x) = \theta_0 + \theta_1 x \quad (5)$$

where x and y represent network complexity and classification loss, respectively. For an illustration of the linear relationship between error rate and parameter size of networks in (5), we can see the experiment in the attached support material. Correspondingly, θ_0 can be estimated by the least-squares method as

$$\theta_1 = \frac{m \sum_{i=1}^m \left(x^i y^i\right) - \sum_{i=1}^m x^i \sum_{i=1}^m y^i}{m \sum_{i=1}^m \left(x^i\right)^2 - \left(\sum_{i=1}^m x^i\right)^2} \quad (6)$$

where $(x^i, y^i) \epsilon \{(x^1, y^1), \ldots, (x^m, y^m)\}$. The red ∇_α in Fig. 1(b) indicates the gradient in the direction of low network complexity and high classification loss of the network. The black ∇_α indicates the gradient in the direction of high network complexity and high classification loss. Therefore, red ∇_α is the required gradient, and black ∇_α is the unnecessary gradient. To distinguish between red ∇_α and black ∇_α, a linear relationship between the network complexity and classification loss is fit by the least square. In the process of updating the architecture parameter gradient ∇_α, the relationship θ_1 between the network complexity and the classification loss can be expressed as

$$\nabla_\theta = \frac{x^k - x^{k-1}}{y^k - y^{k-1}} \quad (7)$$

where x^k and x^{k-1} represent the network complexity at the kth and $(k-1)$th steps, respectively. Similarly, y^k and y^{k-1} represent the classification loss at the kth step and the $(k-1)$th step. When $\nabla_\theta > \theta_1$ or $\nabla_\theta > 0$, the gradient ∇_α is judged to be the red gradient. When $\nabla_\theta < \theta_1$, the gradient ∇_α is judged to be the black gradient.

To simultaneously minimize classification loss and network complexity, the red gradient should be enhanced. Thus, a variable coefficient $(1 + \Phi(\alpha))$ is added to (4) as

$$\nabla'_\alpha \mathcal{L}_{\text{val}} \left(\omega', \alpha\right) - \varepsilon \frac{\nabla'_\alpha \mathcal{L}_{\text{train}} \left(\omega^+, \alpha\right) - \nabla'_\alpha \mathcal{L}_{\text{train}} \left(\omega^-, \alpha\right)}{2\epsilon} \quad (8)$$

where $\nabla'_\alpha = \nabla_\alpha (1 + \Phi(\alpha))$, and $\Phi(\alpha)$ is defined as

$$\Phi(\alpha) = \sigma \frac{|\nabla_\theta|}{\theta_1}. \quad (9)$$

In (9), σ represents the indicating function and is represented as

$$\sigma = \begin{cases} 1, & \text{if } \nabla_\theta > \theta_1 \text{ or } \nabla_\theta > 0 \\ 0, & \text{if } \nabla_\theta < \theta_1. \end{cases} \quad (10)$$

As presented in (5)–(10), when updating the architecture parameters $\alpha^{(i,j)}$, the enhancement of the architecture gradient ∇_α reduces the network complexity and minimizes the classification loss. As we can see from [41] that the gradient in

Algorithm 1 Multi-Objective Evolutionary Optimization Architecture

```
Input: The population size $N$, the maximal generation number $T$, the crossover
probability $\mu$, and the mutation probability $\upsilon$.
1.  Randomly initialize the population $P$ with the size of $N$.
2.  Compute the network complexity $\mathcal{C}(P)$ as the parameter
    size and compute the classification accuracy as [40]
3.  $[F_1, F_2, \ldots] \leftarrow$ Fast non-dominated sorting
    $(\mathcal{L}_{\text{train}}(P), \mathcal{C}(P))$
4.  $[G_1, G_2, \ldots] \leftarrow$ Crowding distance sorting $[F_1, F_2, \ldots]$
5.  **For** $t = 1, 2, 3, \ldots, T$ **do**
6.    Clear offspring population $Q \leftarrow \varnothing$,  $i \leftarrow 0$
7.    **While** $i < N$ **do**
8.        $p \leftarrow (P, [G_1, G_2, \ldots], [F_1, F_2, \ldots])$ select individual by
    the tournament.
9.        $q \leftarrow$ Crossover $(p, \mu)$
10.       $q \leftarrow$ Mutation$(p, \nu)$
11.       $Q \leftarrow Q \cup q; i = i + 1$
12.   **End While**
13.   Compute the network complexity $\mathcal{C}(Q)$ as Eq. (1) and
    compute the classification accuracy $\mathcal{L}_{\text{train}}(Q)$ as [40]
14.  $[F_1, F_2, \ldots] \leftarrow$   Fast non-dominated sorting
    $(\mathcal{L}_{\text{train}}(Q), \mathcal{C}(Q))$
15.  $[G_1, G_2, \ldots] \leftarrow$ Crowding distance sorting $[F_1, F_2, \ldots]$
16.    $P \leftarrow (P \cup Q, [G_1, G_2, \ldots], [F_1, F_2, \ldots])$ select individual
    by the tournament.
17. $t = t + 1$
18. **End for**
    **Output:** The Pareto population $P$
```

the network optimization process based on stochastic gradient descent (SGD) is calculated along the direction of minimizing the loss function. Correspondingly, the proposed algorithm aims to enhance the gradients that tend to obtain a smaller parameter size of network architecture, while leaving the general gradients untouched. Therefore, no treatment is made for the gradients when the indicative function is equal to 0. When the indicative function is 1, it means that the gradient tends to have smaller scale network architecture parameters and the gradient is enhanced. When $\nabla_\theta > \theta_1$ or $\nabla_\theta > 0$, the gradient tends to obtain smaller scale network parameters size and high performance, and the gradient is enhanced.

D. Network-Level Search Strategy

Through the first stage (blocks-level search stage), we have obtained the normal block and reduction block. However, the shared blocks need to be optimized because of the gap between the proxy network and the target network. Therefore, when building target networks, a multiobjective evolutionary optimization algorithm NSGA-II [24] is used. The depth and width of the target network will be the search space at this stage. The detailed configuration of the search space in this stage will be introduced in Section IV-A. For the design of the spatial resolution in the target network, we adopt the one used in [23] and [31].

At the network-level search stage, the algorithm is an iterative process. In the search process, the initial architecture as a group is improved gradually. At each iteration, a set of offspring is created by applying the mutations and crossover operations. Fig. 1(d) shows an example of applying this Algorithm 1 to searching on CIFAR10. Because the resolution of CIFAR samples is 32×32, the target network consists of two downsampling layers and the final stage of the target network is a fully connected layer. As the example of searching architecture for CIFAR10, the search space of the multiobjective evolutionary Algorithm 1 contains six variables. We use a mixed encoding of real and integer variables to accommodate the construction of a target network. Integer variables $\upsilon_1^1, \upsilon_3^1, \upsilon_5^1$ represent the size of the layers with normal blocks between reduction blocks. Real variables $\upsilon_2^1, \upsilon_4^1$ represent the increment of the channels when the feature is downsampling. The integer variable υ_0^1 indicates the size of the first convolutional layer channels.

The purpose of optimizing the architecture is to obtain the network architecture with low complexity and excellent performance. However, if we use $h = (\mathcal{L}_{\text{val}}(\omega^*(\alpha), \alpha)/\mathcal{C}(\alpha))$ to measure the value of individuals in the Pareto set, the individual $\upsilon_{h_{\max}}$ with the maximum value h cannot achieve the goal of optimizing the architecture. Instead, we need the best individual as α_{best} in Fig. 2. Thus, a decision-making method is designed to obtain the optimal neural network architecture.

The proposed decision-making method is implemented by measuring the distance d from the individual to the linear function $Ax + y + B = 0$, and the optimal individual is the one with the largest distance $d_{\max}$. The specific algorithm is shown in Algorithm 2. The optimal individual selected by this method has taken into account the performance and complexity of the neural network architecture simultaneously.

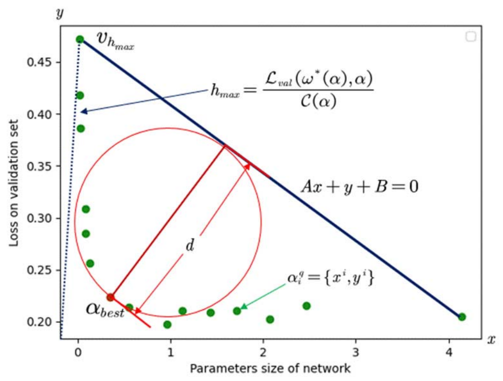

Fig. 2. Schematic of the decision-making method for the optimal neural network architecture. The green points represent individuals in the Pareto set. By calculating the distance of each individual to the linear function $Ax + y + B = 0$, the optimal individual is obtained.

IV. Experimental Setup and Result

In this section, we first present the implementation details of our experiments. Then, we discuss the effectiveness of the block-level search process and the properties of the Pareto set. Next, we analyze the training process of block-level search and network-level search on CIFAR10 and CIFAR100. Finally, we analyze ablation experiments about the parameter settings of the proposed framework and the effectiveness of algorithm components.

A. Implementation Detail

In order to demonstrate the effectiveness of the proposed algorithm, we compare the Pareto set of EG-DARTS with those produced by other state-of-the-art methods. Based on the ease of use and efficiency of algorithm comparison, these methods are compared on the CIFAR datasets.

CIFAR10 is an image classification dataset composed of ten classes of objects, including training data of 50 000 images and test data of 10 000 images. Moreover, the size of each image in CIFAR10 is 32×32 pixels. The image quality of CIFAR100 is roughly the same as that of CIFAR10, except that it is composed of 100 categories. Similarly, in CIFAR100, the number of images for train data is 50 000 and the number of images for test data is 10 000. Hence, the number of images contained in each category of CIFAR100 is one-tenth of that of CIFAR10. In addition, we split the CIFAR10 and CIFAR100 training sets according to (80%–20%) to construct the training and validation sets to prevent overfitting and improve generalizability.

Moreover, the comparison methods we select are mainly categorized into four groups: the first group contains architectures designed by human experts including ResNet and DenseNet. The second group consists of single-objective NAS methods based on EA or gradient-based, which have been proven to be effective in completing various computer version tasks. Another group is the model compression which is to show the compression efficiency of the proposed algorithm.

Algorithm 2 Decision-Making Approach for the Optimal Neural Network Architecture

```
Input: Pareto set $P_q = \{\alpha_1^q, \alpha_2^q, \ldots, \alpha_i^q, \ldots, \alpha_n^q\}$, where $\alpha_i^q = \{x^i, y^i\}$.
1. Based on the point $(x_0^{\max}, y_0)$ with the largest parameter size of the network architecture and the point $(x_1^{\min}, y_1)$ with the smallest parameter size, the linear function $Ax + y + B = 0$ is constructed, where $A = -\frac{y_0 - y_1}{x_0^{\min} - x_1}$ and $B = -\frac{y_0 - y_1}{x_0^{\max} - x_1^{\min}} x_0 - y_0$.
2. $S_d \leftarrow \varnothing$
3. For $i = 1, 2, 3, \ldots, n$ do
4.     Calculate the distance $d_i = \frac{Ax^i + y^i + B}{A^2 + 1}$
5.     $S_d \leftarrow S_d \cup d_i$
6. End for
7. Get the $\alpha_{best} = \arg \max_{\alpha(x^t, y^t)} (S_d)$
Output: The optimal individual $\alpha_{\text{best}}$.
```

The final group is multiobjective NAS which is to compare our algorithms more fairly. In the literature, various metrics can represent the complexity of the network, including the parameter size, the inference time, and the number of FLOPS. Due to the difference and inconsistency of factors such as computing environment and ambient temperature, the inference time may not be reliably used as a network complexity metric. Therefore, we choose parameter size and FLOPS as the indicators of network complexity.

In the process of the block-level stage, we use the standard SGD method to learn the weights of the network and the weights of the architecture parameters. In the network-level search process, the SGD is used to estimate the classification accuracy of each corresponding to the network architecture. At the same time, NSGA-II is used to design low-complexity and high-performance target networks. To make the search process more efficient and meet the search requirements, the computational resource consumption of the algorithm needs to be reduced as much as possible. Table I summarizes the hyperparameter settings in EG-DARTS related to search space, gradient descent training, and search strategy. All the experiments are conducted on two GPUs with NVidia GeForce RTX TITAN and two CPUs with Intel Xeon Silver 4214. The codes of EG-DARTS are coded by Python 3.6.9 and Pytorch 1.7.

The training setup for block-level search largely follows [25], and the training setting for network-level search mainly follows [11]. For the block-level search, the training process is divided into three stages, the number of epochs in each stage is set to 40, and the batch size is set to 96. For the network-level search, each network architecture is trained with 30 epochs, and the batch size is 128 and completely retrained from scratch. The obtained network architecture is completely retrained from scratch. During the training process, the number of epochs is extended to 600, the batch size is 128, and we introduce the data preprocessing technique CutOut [42]. To further optimize the training process, an auxiliary head classifier is added to the network [2]. The loss of this auxiliary head classifier is scaled by a constant factor of 0.4 and is aggregated with the loss of the original structure during the training process before backpropagation.

TABLE I
SUMMARY OF HYPERPARAMETER SETTING

Categories	Parameter name	Parameter value
Gradient Descent (Block-Level)	batch size	96
	weight decay	5.00E-04
	training epochs	40
	learning rate	0.025
Gradient Descent (Block -Level)	batch size	128
	weight decay	5.00E-04
	training epochs	36/600
	learning rate	0.025 with Cosine Annealing
Search Space (Network-Level)	v_0	(8,60)
	v_1	(1,6)
	v_2	(1,6)
	v_3	(1,6)
	v_4	(1,3)
	v_5	(1,3)
Search Strategy (Network-Level)	population size	15
	generations	20
	crossover probability	0.9
	mutation probability	0.1

To verify the effectiveness of the proposed EG-DARTS approach, a series of experiments is designed and performed. The designed experiments can be divided into two different categories. The first category is comparison experiments. The manual neural network architectures that are hand-crafted such as ResNet and DenseNet, are selected to compare with EG-DARTS. The neural network architecture search algorithms such as single-objective NAS, multiobjective NAS, and model compression are also chosen for comparison. The second experiment category covers the performance analyses of different parameters and components of EG-DARTS and further explanation of the impact of some important hyperparameters on the performance with designed neural networks.

B. Efficiency of EG-DARTS

In the literature, the parameter size of the network architecture is used to measure the complexity of the network architecture for most of the neural architecture searching approaches on the CIFAR. Here, to prevent potential discrepancies from reimplementation, we also use the parameter size to compare the complexity of the network. Inspired by the training process presented in [25], CutOut is utilized for data augmentation to facilitate comparison with other peer algorithms. For comparison purposes, we analyze and compare the proposed algorithms in terms of manual design, single-objective NAS, model compression, and multiobjective NAS, respectively, as shown in Table II.

In Table II, EG-DARTS-A0, EG-DARTS-A1, and EG-DARTS-BEST represent the search results of the proposed approach on CIFAR10 and CIFAR100. EG-DARTS-A0 and EG-DARTS-A1 are chosen to facilitate a fair comparison with NAS using non dominated sorting genetic algorithm-II

TABLE II
COMPARISON OF IMAGE CLASSIFICATION ARCHITECTURES ON CIFAR10 AND CIFAR100

Categories	Algorithms	Test Err. (%)		#Params (M)	GPU Days	Search Method
		CIFAR10	CIFAR100			
Manual	DenseNet-BC(k=49,depth=190) [59]	3.46	17.18	25.6	-	manual
	ResNet(depth=110) [3]	7.93	25.16	1.70	-	manual
Single-Objective NAS	AmoebaNet-A+CutOut [43]	3.34	-	3.20	3150	EA
	AmoebaNet-B+CutOut [43]	2.55	-	2.80	3150	EA
	Hireachical Evolution [44]	3.75	-	15.70	300	EA
	Genetic CNN [27]	7.10	29.05	-	14	EA
	Block-QNN-S [12]	4.38	20.65	6.1	90	RL
	CGP-CNN [45]	5.98	-	2.64	27	EA
	PNAS [31]	3.41	-	3.20	255	SMBO
	ENAS+CutOut [46]	2.89	-	4.60	0.5	RL
	DARTS(first order)+CutOut [23]	3.00	17.76	3.30	1.5	gradient-based
	DARTS(second order)+CutOut [23]	2.76	17.54	3.30	4	gradient-based
	AE-CNN+E2EPP [20]	5.30	22.02	4.3/20.9	8.5	EA
	SI-EvoNet [47]	4.98	21.84	0.51/0.99	0.8	EA
Model Compression	N2N(Student) [48]	8.36	31.99	0.98/2.42	-	architecture compression
	AMC [49]	6.45	-	9.4	-	architecture compression
	BO-KD(C10/C100) [50]	7.73	26.17	0.81/1.87	-	knowledge distillation
	ResKD(C10/C100) [51]	6.73	28.04	0.44/0.45	-	knowledge distillation
	Self-Distillation(C10/C100)[52]	6.32	26.78	0.5/0.9	-	knowledge distillation
Multi-Objective NAS	MOO-EC(C10/C100) [53]	8.70	26.63	0.49/1.1	-	EA
	DDP-Net(C10) [37]	4.62	-	0.5	-	sequential model-based
	Proxyless-G+CutOut [54]	2.08	-	5.7	-	others
	LEMONADE [55]	4.57	-	0.5	-	EA
	MOGIG-Net [56]	4.67	24.71	0.9/0.7	-	EA
	MOE-CL [38]	11.10	-	0.53	-	EA
	MDARTS [57]	2.56	17.22	2.1/1.9	0.50	gradient-based
	MOPSO/D-Net [39]	5.88	-	8.1	0.30	PSO
	NSGANet [58]	2.75	-	3.3	4	EA
	NSGANetV1-A0 [11]	4.67	25.17	0.2/0.2	27.00	EA
	NSGANetV1-A1 [11]	3.49	19.23	0.5/0.7	27.00	EA
Ours	EG-DARTS-A0+CutOut	4.52	22.00	0.24/0.35	7	gradient-based + EA
	EG-DARTS-A1+CutOut	3.47	19.16	0.50/0.97	7	gradient-based + EA
	EG-DARTS-BEST+CutOut	3.18	22.00	0.64/0.35	7	gradient-based + EA

(NSGANetV1)-A0 and NSGANetV1-A1 [11], respectively. The networks of EG-DARTS-A0 are with fewer parameters, which are convenient for comparison with NSGANetV1-A0. The EG-DARTS-A0 on CIFAR10 is with 0.24-M parameters and a 4.52% error rate. Equally, the EG-DARTS-A0 on CIFAR100 is with 0.35-M parameters and a 22.0% error rate. Similarly, EG-DARTS-A1 represents two networks that are obtained through EG-DARTS on CIFAR10 and CIFAR100, and they are also convenient for comparison with NSGANetV1-A1. The EG-DARTS-A1 on CIFAR10 is with 0.5-M parameters and a 3.47% error rate. Likely, the EG-DARTS-A1 on CIFAR100 is with 0.97-M parameters and a 19.16% error rate.

Compared with the manually designed classical network architecture, EG-DARTS is the most similar to ResNet (depth = 110) [3] in terms of the number of parameters. On CIFAR10, EG-DARTS-A1 is not only 3.5× smaller in parameter size than ResNet(depth = 110) but also lower than ResNet(depth = 110) in error rate. On CIFAR100, EG-DARTS-A1 is slightly smaller than ResNet(depth = 110) in parameter size, and EG-DARTS-A1 is lower than ResNet(depth = 110) in error rate. The classification error rate of EG-DARTS-A1 is slightly higher than that of DenseNet-BC ($k = 49$ and depth = 190) on CIFAR10, and EG-DARTS-A1 is higher than that of DenseNet-BC ($k = 49$ and depth = 190) on CIFAR-100. However, the parameter size of DenseNet-BC ($k = 49$ and depth = 190) is 25× more than EG-DARTS-A1. Through the above comparison, it can be seen that EG-DARTS obtains obvious performance advantages over manual network architecture in the low parameter size architecture.

Accordingly, we also compared the single-objective NAS competitors [12], [20], [23], [27], [31], [43], [44], [45], [46], [47] of the last few years. The single-objective NAS algorithms focus more on the network architecture performance improvement, so the architecture parameter size is mostly larger than 1 M. In the cases of competitors with parameter sizes larger than 1 M, EG-DARTS is disadvantaged and only has some advantages over Genetic CNN [27], block-wise neural network architecture by Q-learning framework (Block-QNN-S) [12], Cartesian genetic programming (CGP)-CNN [45], and AE-CNN combined with present an effective and efficient end-to-end performance predictor (AE-CNN+E2EPP) [20] in terms of classification error rate. The latest single-objective NAS algorithm SI-EvoNet builds a low parameter size network architecture and achieves excellent performance, so we compare it with SI-EvoNet in detail. On CIFAR10, the parameter size of EG-DARTS-A1 is slightly lower than that of SI-EvoNet and the classification error rate is 1.51% lower than that of SI-EvoNet. Similarly, on CIFAR100, the parameter size of EG-DARTS-A1 is slightly lower than that of SI-EvoNet while the classification error rate is lower than that of SI-EvoNet. Compared with SI-EvoNet, EG-DARTS-A1 has a lower parameter size and better classification accuracy.

Because our proposed algorithm considers both classification error rate and parameter size, we also compare related

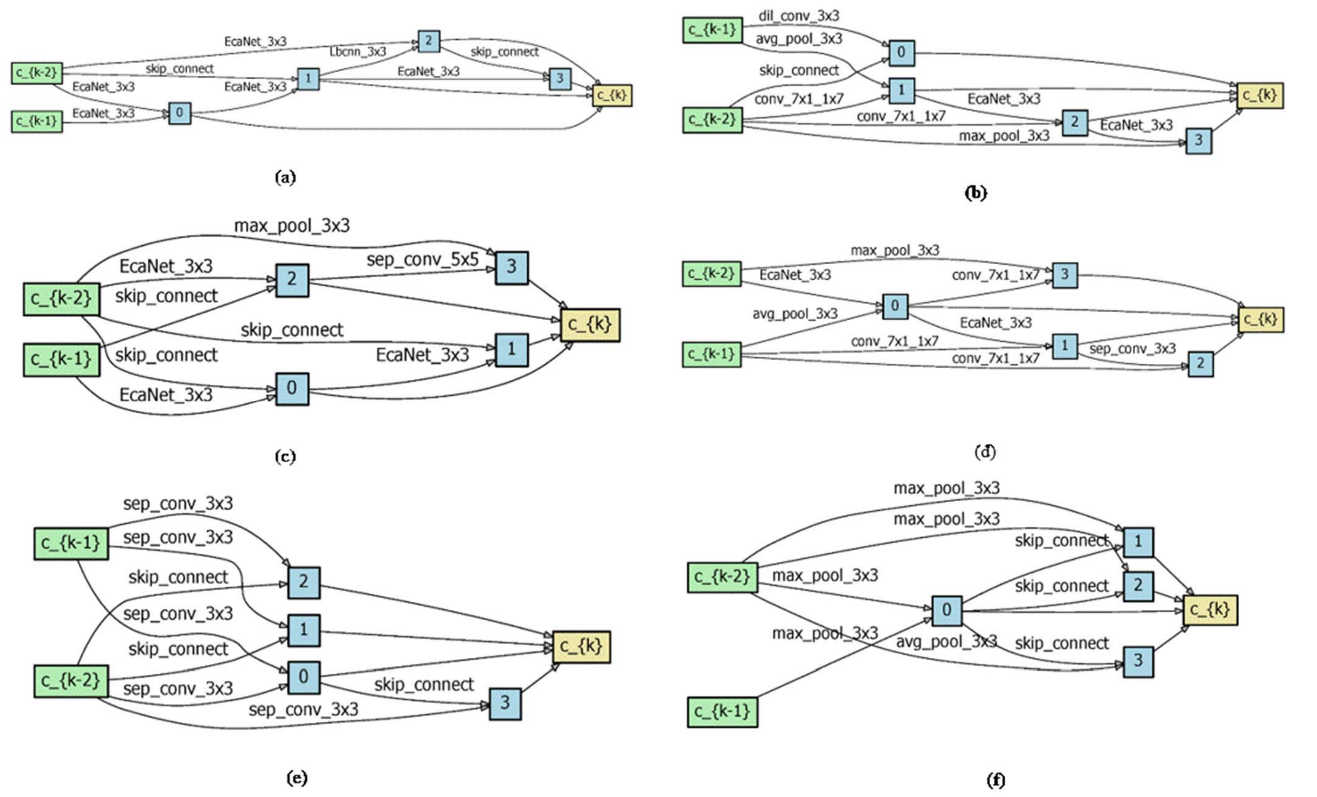

Fig. 3. Block architecture learned on different classification tasks: (a) and (b) represent the normal block and the reduction block, respectively, learned on CIFAR10 by EG-DARTS; (c) and (d) represent the normal block and the reduction block, respectively, learned on CIFAR100 by EG-DARTS; and (e) and (f) represent the normal block and the reduction block, respectively, from [23] on CIFAR10.

algorithms [48], [49], [50], [51], [52] with model compression to test the efficiency of the algorithm. The comparison shows that EG-DARTS obtains a lower classification error rate when the parameter size is below 1 M. Among the competitors of the compared model compression, self-distillation [52] architecture parameter size is most similar to the proposed algorithm. Further, the classification error rate of EG-DARTS-A1 is 1.8% and 4.78% lower than self-distillation [52] on CIFAR10 and CIFAR100, respectively. Therefore, EG-DARTS has higher compression efficiency and performance than the model compression algorithms.

Finally, we also compare multiobjective NAS algorithms [37], [38], [39], [53], [54], [55], [56], [57], [58] within the last few years, which all consider both the complexity of the network architecture and the classification error rate. On CIFAR10, EG-DARTS has disadvantage (e.g., Proxyless-G+CutOut [54] and MDARTS [57]) in the case of parameter size larger than 1 M and only has a lower error rate than MOPSO/D-Net [39] by 1.36%. However, EG-DARTS shows excellent performance when the competitor architecture parameter size is less than 1 M. On CIFAR10, EG-DARTS-A1 is 7.63% lower than that of MOE-CL [38]. On CIFAR10, EG-DARTS-A1 with the same parameter size is 1.1% lower than LEMONADE [55] in the classification error rate. NSGANetV1 [11] is the strongest competitor among the comparative algorithms for multiobjective NAS. On CIFAR10, EG-DARTS-A0 with 0.2 M parameter size is 0.15% lower than NSGANetV1-A0 in classification error rate. At the parameter size is about 0.5 M, EG-DARTS-A1 is slightly lower than NSGANetV1-A1 in error rate. Correspondingly, on CIFAR100, EG-DARTS-A1 not only obtains 7.27% lower than multi-objective optimization by evolutionary computation (MOO-EC) [53] on classification error rate but also obtains 0.13 M lower than it on parameter size. Equally, NSGANetV1 is still the most favorable competitor on CIFAR100, and EG-DARTS-A0 obtains 3.17% lower than NSGANetV1-A0 [11] in classification error rate as the case of parameter size is about 0.2 M. When parameter size is less than 1 M, EG-DARTS-A0 shows a lower error rate than the competitors on CIFAR100. Thus, we are able to conclude that the network architecture constructed by EG-DARTS obtains a lower classification error rate than other multiobjective NAS algorithms at parameter size below 1 M.

Nevertheless, there are still some weaknesses of the network architecture obtained by EG-DARTS that need to be further investigated and explored. First, the performance of the obtained network architectures with larger parameter sizes needs to be further improved, and the import of some new algorithms may further enhance the performance. In addition, we find that high-performance network architectures with low parameter sizes have relatively long inference times for computation, which can be seen in Section IV-D. This issue can be further addressed in future studies.

C. Block-Level Search Analysis

The block topology can be obtained using the approach of enhancing gradient by searching in a proxy network, which is consisted of the initial stage, the intermediate stage, and the final stage. At the initial stage, the proxy network depth is five, and there are 12 operations between two nodes. According

TABLE III

(A) PARETO SETS SEARCHED ON CIFAR10 BASED ON THE BLOCKS OF FIG. 3(A) AND (B). (B) PARETO SET SEARCHED BY THE APPROACH ON CIFAR100 BASED ON THE BLOCKS OF FIG. 3(C) AND (D). #PARAMS REPRESENTS THE PARAMETER SIZE OF THE NETWORK. ERR. REPRESENTS THE CLASSIFICATION ERROR RATE OF THE NETWORK. FLOPS IS THE NUMBER OF FLOATING-POINT OPERATIONS OF THE NETWORK. LATENCY REPRESENTS THE AVERAGE INFERENCE TIME CONSUMED BY A SINGLE IMAGE INFERENCE OBTAINED BY REPEATING IMAGE INFERENCE 1000 TIMES ON THE COMPUTER WITH A GPU OF 1080TI. DEPTH IS THE NUMBER OF CNN BLOCK LAYERS

(a)

v_0	v_1	v_2	v_3	v_4	v_5	#Params (M)	Err.(%)	Err.(%) +CutOut	FLOPS （M）	Latency（ms）	Depth
50	6	2	2	2.50	1.52	2.26	3.17	2.81	541.161	25.83	10
35	4	3	2	2.50	1.70	1.29	3.42	3.23	259.59	23.65	9
35	4	3	2	2.50	1.16	0.89	3.3	3.18	228.38	23.65	9
27	2	3	2	2.42	1.79	0.75	3.66	3.61	130.72	19.41	7
50	**6**	**2**	**3**	**1.20**	**1.09**	**0.64**	**3.19**	**3.18**	**305.32**	**27.92**	**11**
27	4	3	2	1.36	2.84	0.50	3.99	3.58	99.56	23.64	9
28	2	2	2	1.31	1.83	0.28	4.25	4.36	64.44	17.32	6
28	2	2	2	1.20	1.83	0.24	4.08	4.52	58.27	17.3	6
16	2	3	3	1.87	1.09	0.14	5.09	4.99	32.45	21.54	8
11	3	3	2	2.50	1.68	0.13	5.24	5.41	25.64	21.53	8
9	6	4	4	2.33	1.28	0.09	5.28	5.85	22.63	34.47	14
9	6	4	4	1.85	1.28	0.06	6.34	6.87	18.04	34.44	14
15	2	2	3	1.19	1.28	0.06	6.75	7.07	16.83	19.42	7
9	6	4	4	1.41	1.19	0.04	7.8	8.68	15.01	34.44	14
9	6	4	2	1.41	1.16	0.03	7.79	9.11	14.72	30.12	12

(b)

v_0	v_1	v_2	v_3	v_4	v_5	#Params (M)	Err.(\%)	Err.(%) +CutOut	FLOPS （M）	Latency （ms）	Depth
54	6	2	2	2.93	2.42	4.14	20.50	19.03	679.43	13.60	10
54	6	2	2	2.80	1.76	2.47	21.54	19.46	531.07	13.22	10
46	6	2	2	2.41	2.38	2.07	20.22	19.55	383.32	13.21	10
51	6	2	2	2.61	1.61	1.72	21.09	19.54	409.62	13.22	10
56	6	2	3	2.29	1.15	1.43	20.90	19.34	417.36	14.27	11
56	6	2	2	2.29	1.03	1.12	21.07	20.00	395.41	13.19	10
48	6	3	4	1.12	2.38	0.97	19.72	19.16	244.62	16.26	13
44	6	3	2	1.08	2.46	0.55	21.42	20.97	188.23	14.24	11
35	**6**	**3**	**2**	**1.07**	**2.45**	**0.35**	**22.40**	**22.00**	**116.16**	**14.22**	**11**
22	3	6	2	1.17	1.79	0.13	25.62	26.40	37.18	14.25	11
14	2	2	3	1.76	1.45	0.08	28.50	29.94	16.13	10.08	7
10	5	2	2	1.98	2.37	0.08	30.90	30.28	15.11	12.14	9
8	5	2	3	1.96	1.03	0.03	38.61	40.47	8.50	13.20	10
8	2	2	2	1.83	1.12	0.02	41.83	44.50	5.44	9.07	6
8	5	2	3	1.14	1.11	0.02	47.16	49.91	6.73	13.19	10

to the order of the architecture parameter, four operations with the smallest are deleted. At the intermediate stage, the depth of the trained proxy network is 11, and there are eight operations between two nodes. Once this stage is completed, only four operations are left between the nodes. At the final stage, the depth of the trained proxy network is 17, and there are four operations between two nodes. What we can obtain from the end stage is the final block topology where each node is connected by a remaining operation and the other three operations are deleted with the minimum three architecture parameters α. The block topology diagram obtained from the block-level search is shown in Fig. 3.

Fig. 3(a) and (b) shows the block topology searched on CIFAR10, where Fig. 3(a) shows the structure of the normal block and Fig. 3(b) shows the structure of the reduction block. The resolution of the feature map remains unchanged in the normal block but the resolution is reduced to a half in the reduction block. Therefore, there will be a large number of downsampling operations between nodes in the reduction block. Most operations in the normal block are convolution ones to keep the resolution of the data feature map unchanged while further extracting the features of the data information. There are more sep_conv_3 $\times$ 3 operations in these blocks as demonstrated in Fig. 3(e) and (f), compared with DARTS [23]. On the other hand, the block structure Fig. 3(a) and (b) obtained by our approach contains more EcaNet_3 $\times$ 3 operations. It has been revealed in [6] and [60] that the main purpose of the sep_conv_3 $\times$ 3 operation is to reduce the size of parameters, and the EcaNet_3 $\times$ 3 operation is a module of the attention mechanism, which reduces operation parameter size while obtaining better feature extraction capabilities. Therefore, the EcaNet operations are the main part of the blocks. Fig. 3(c) and (d) demonstrates the search results on CIFAR100. Compared with the search results on CIFAR10, skip-connect operations appear more times in the block structure of Fig. 3(c) and (d). The reason for the block structure on CIFAR100 may be related to its data distribution. Compared with 6000 images in each class of CIFAR10, there are only 600 images in each class of CIFAR100. The skip connection in the blocks is to avoid network overfitting.

D. Network-Level Search Analysis

Based on the blocks obtained from the block-level search, the target network is constructed by the network-level search. Table III shows the set of Pareto solutions obtained by the network-level search on CIFAR10 and CIFAR100. The

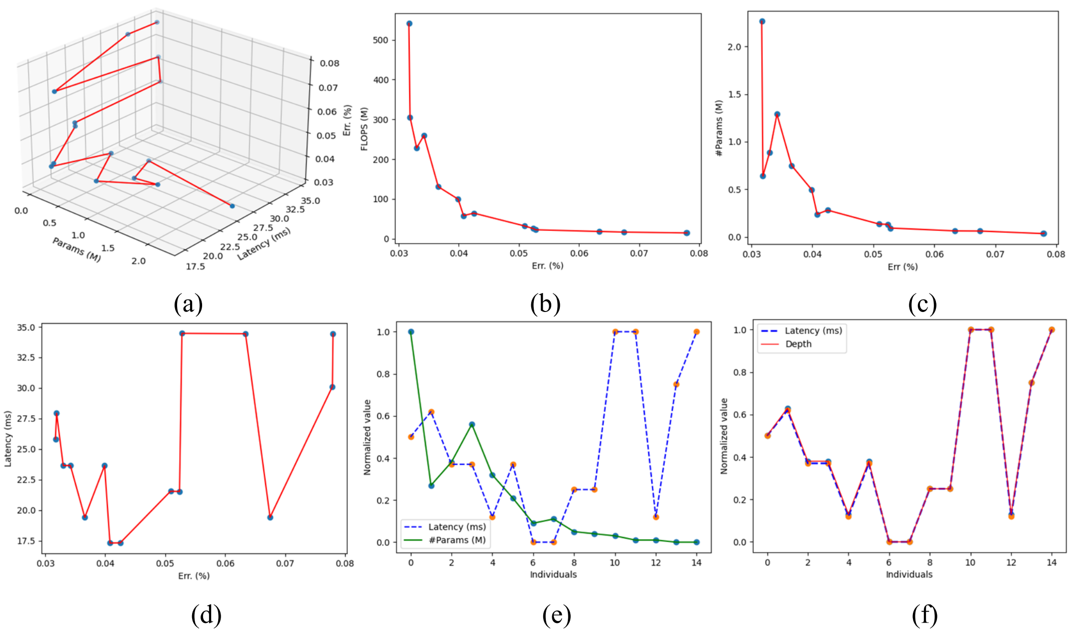

Fig. 4. Characteristic relationship diagram of the obtained Pareto set on CIFAR10. The diagram is designed to present the relationship between accuracy, network complexity, and inference time: (a) 3-D diagram of the relationship between classification error rate (Err.), network parameter size (#Params), and inference time (Latency); (b) Pareto front with the classification error rate and the number of FLOPS; (c) relationship between the classification error rate and parameter size; (d) relationship between the classification error rate and inference time; (e) individuals of the Pareto set are normalized on the parameters size and the inference time; and (f) individuals of the Pareto set are normalized on the network depth and the inference time.

network-level architecture parameters υ_0–υ_5 are obtained by searching on the constraint of search space (Table I).

According to the proposed decision method (Algorithm 2), the data in the fifth row of Table III(a) show the optimal network-level architecture obtained on CIFAR10 (referred to as decision individual A), and the ninth row of Table III(b) shows the optimal network-level architecture obtained on CIFAR100 (referred to as decision individual B). In Table II, EG-DARTS-BEST represents Individual A searched on CIFAR10 and Individual B searched on CIFAR100. EG-DARTS-BEST on CIFAR10 is with 0.64-M parameters and a 3.18% error rate. EG-DARTS-BEST obtains the lowest error rate among models with parameter size below 0.7 M, while EG-DARTS-BEST on CIFAR100 is with 0.35-M parameters and a 22.00% error rate. Similarly, EG-DARTS-BEST on CIFAR100 obtains the lowest error rate among models with parameters below 0.5 M. Since the Pareto set contains high-performance individuals with low parameter sizes, its architectural parameters are useful for guiding the design of the network architecture. Some patterns can be analyzed from the architectural parameters in Table III. The architecture parameter of decision individual A is 4 and larger than υ_2 and υ_3, and the architecture parameter υ_1 of decision individual B is 6 and larger than υ_2 and υ_3. The architecture parameter υ_1 for individuals with an error rate below 3.5% in Table III(a) is larger than υ_2 and υ_3. Similarly, the architecture parameter υ_1 for individuals with an error rate below 22.0% in Table III(b) is larger than υ_2 and υ_3. The possible reason is that the network architecture near the input side of the image requires more convolutional layers to filter the noise in the data and extract the features. υ_2 and υ_3 are relatively small which may be that part of the feature map already contains the features of the data. Therefore, only a small number of convolutional layers are needed to fit the distribution of the data. The individual in the fifth row in Table III(a) has a parameter size of 0.64 M and an error rate of 3.18%, which obtains high performance. However, the FLOPS and inference time of this individual are both the highest in the Pareto set. Similarly, the individual in the seventh row of Table III(b) obtains a low parameter size and low error rate, but its FLOPS and inference time are relatively highest. It shows that the complexity of the network is not linearly related to the latency. Next, we analyze the correlation between latency and network architecture, as shown in Figs. 4 and 5.

Fig. 4(a) shows the relationships among the classification error rate, network parameter size, and inference time of the Pareto set for CIFAR10, where the data of the inference time are more discretized. Fig. 4(b) shows the Pareto front obtained by EG-DARTS. Fig. 4(c) is consistent with Fig. 4(b), which shows that the network architecture attributes represented by FLOPS are similar to those represented by the parameter size of the network. Fig. 4(d) is quite different from that in Fig. 4(b), which further suggests that the inference time is not linearly related to the parameter size. Similarly, the abovementioned data distribution also appear on CIFAR100, as shown in Fig. 5(a)–(d).

To further study which network architecture attributes affect the network inference time, we construct the trend curve based on network parameter size, network depth, and network inference time normalized data, as shown in Figs. 4(e) and (f) and 5(e) and (f). It can be concluded that the inference time of the network has the strongest correlation with the depth of the network, and the curve trends of the inference time and network depths are almost coincident. This suggests that when the data are calculated in the GPU, the inference time is mainly generated by the feature map

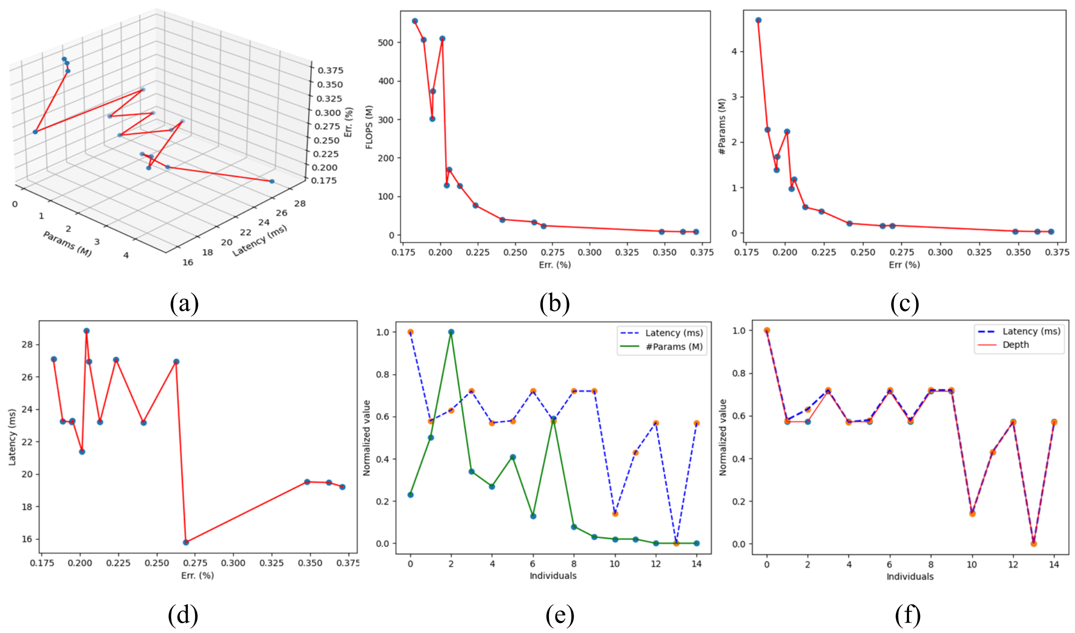

Fig. 5. Characteristic relationship diagram of the Pareto set on CIFAR100: (a) 3-D diagram of the relationship among classification error rate, network parameter size, and inference time; (b) Pareto front with the classification error rate and the number of FLOPS; (c) diagram of the relationship between the classification error rate and parameter size; (d) diagram of the relationship between the classification error rate and inference time; (e) individuals of the Pareto set are normalized on the parameter size and the inference time; and (f) individuals of the Pareto set are normalized on the network depth and the inference time.

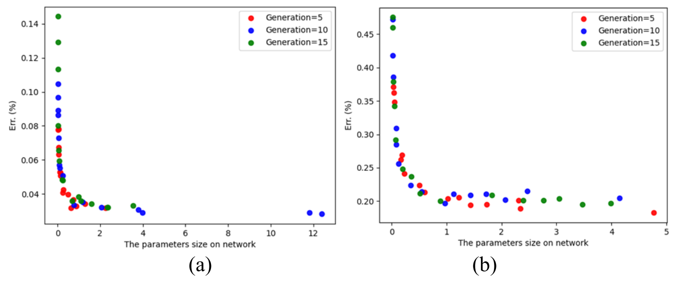

Fig. 6. Pareto front with the classification error rate and parameter size when the number of generations is set as 5, 10, and 15: (a) CIFAR10 and (b) CIFAR100.

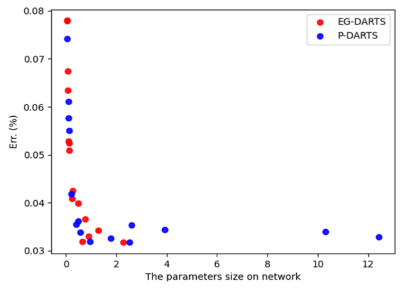

Fig. 7. Pareto front with the classification error rate and parameter size on the blocks through P-DARTS [25] and EG-DARTS.

through each layer of convolution. The channels of each layer affect the amount of data-parallel calculation and do not affect the inference time. To summarize, the parameter size of the network and the number of FLOPS represent the same network attributes and the inference time cannot be expressed by the parameter size and the number of FLOPS. The inference time is linearly related to the depth of the network.

E. Ablation Study

Fig. 6 shows the Pareto set under different generations on both CIFAR10 and CIFAR100. It can be seen from Fig. 6(a) that the increase of generation on CIFAR10 has little effect on the approximate Pareto front boundary. This phenomenon indicates that the number of generations is set as 5 based on CIFAR10, and EG-DARTS has converged. Fig. 6(b) is the ablation experimental result on CIFAR100. When the number of generations increases, the front boundary of Pareto does not change significantly. As described, the number of generations is set to 5, and it can converge on CIFAR10 and CIFAR100.

For further analysis, the blocks of P-DARTS are utilized for network-level search. For the fairness of the experiment, we utilize the same training pipeline without data augmentation (CutOut) in Table III on CIFAR10. The solution set distribution is shown in Fig. 7. We compare the parameter size by selecting individuals with the same error rate from the solution sets based on the P-DARTS blocks and the EG-DARTS blocks, as shown in Table IV. Individuals based on the EG-DARTS blocks obtain lower parameter size. This experiment clarifies that the block structure obtained by EG-DARTS contains a lower parameter size in the case of the same performance. To further illustrate the impact of the search of the block topology on the target network, we apply the network-level parameters in Table III(a) and the P-DARTS block to construct the target network (that is, fixing v_0–v_5) for experiments with the same training pipeline as in Table III and without using data augmentation (CutOut). The experimental results

TABLE IV

COMPARISON OF IMAGE CLASSIFICATION PERFORMANCE FOR TARGET NETWORK WHICH BLOCKS PRODUCED BY P-DARTS AND EG-DARTS

Block search method	#Params (M)	Err. (%)
P-DARTS Blocks	0.96	3.19
EG-DARTS Blocks	**0.64**	**3.19**

TABLE V

COMPARISON OF PARETO SET FOR TARGET NETWORK ON P-DARTS BLOCK AND EG-DARTS BLOCK WITH THE SAME NETWORK-LEVEL ARCHITECTURE

EG-DARTS Block		P-DARTS Block	
#Params (M)	Err.(%)	#Params (M)	Err.(%)
2.26	**3.17**	2.83	3.46
1.29	**3.42**	1.60	3.69
0.89	**3.3**	1.10	3.58
0.75	**3.66**	0.92	4.05
0.64	**3.19**	0.80	3.49
0.50	**3.99**	0.61	4.03
0.28	**4.25**	0.34	4.59
0.24	**4.08**	0.28	4.42
0.14	**5.09**	0.16	5.11
0.13	**5.24**	0.15	5.28
0.09	**5.28**	0.11	5.4
0.06	**6.34**	0.07	6.35
0.06	6.75	0.07	6.62
0.04	7.8	0.04	7.69
0.03	**7.79**	0.04	7.86

are shown in Table V. It can be seen from Table V that with the same network-level parameters (v_0–v_5), the EG-DARTS blocks-based target network obtains lower parameters with higher performance. When the parameter size of the network architecture is less than 0.1 M, the parameter size of the network has a greater impact on performance. Therefore, the network based on the P-DARTS block has a lower classification error rate with a larger parameter size, as shown in the data in the 13th and 14th rows of Table V. Through the above experiments, it can be concluded that the block structure from EG-DARTS block-level search has lower parameters and better performance.

To further illustrate the difference between P-DARTS and EG-DARTS, the 2-D Kolmogorov–Smirnov test (2-D KS-Test) is utilized to differentiate the performance of P-DARTS and EG-DARTS. The experimental details can be found in Section II of the attached support material. According to the KS-Test results in the attached support material, it can be seen that there is a significant difference between the model based on the EG-DARTS block and the model based on the P-DARTS block. Based on the experimental results in Tables IV and V, we infer that the performance of the EG-DARTS block is better than that of the P-DARTS block.

To further analyze the role of network-level search for EG-DARTS, we conducted experiments on CIFAR10 with uniformly distributed network-level architectural parameters based on EG-DARTS blocks, which generated a total of 25 samples. The results are shown in Table VI. The same training pipeline as in Table III is utilized for the samples and no data augmentation (CutOut) was applied. From the experimental data in Table VI, it can be seen that the classification error rate for all 25 samples reached above 4%. The experiment shows that EG-DARTS block-based and randomly set network-level parameters cannot build network architectures with a lower size of parameters with high performance. The network-level search method of EG-DARTS is necessary to build high-performance target networks with a lower size of parameters.

Since individuals with low parameter amounts and high performance have been obtained, we consider whether it is

TABLE VI

COMPARISON OF TARGET NETWORK WITH UNIFORMLY DISTRIBUTED NETWORK-LEVEL ARCHITECTURAL PARAMETERS BASED ON EG-DARTS BLOCKS

v_0	v_1	v_2	v_3	v_4	v_5	#Params(M)	Err.(%)
56	2	3	2	2.74	2.53	21.31	4.32
55	4	1	5	2.67	2.54	16.85	4.34
71	3	1	1	2.58	2.34	10.37	4.79
50	3	5	4	1.99	2.96	9.52	4.22
40	2	5	4	2.31	2.81	7.70	4.17
70	1	4	4	1.38	2.54	6.91	4.51
48	4	2	4	2.65	1.88	6.88	4.26
76	3	2	3	1.74	1.90	6.76	4.47
64	4	1	3	2.12	1.83	6.22	4.79
51	5	4	1	2.99	1.68	5.53	4.25
47	4	3	1	1.99	2.91	4.12	4.78
66	2	4	1	1.89	1.27	3.03	4.47
55	5	1	2	1.55	2.28	2.91	4.62
34	1	5	2	2.69	1.81	2.77	4.17
49	4	4	3	1.62	1.86	2.45	4.18
54	1	5	1	2.06	1.13	2.32	4.43
58	3	2	4	1.04	1.61	1.41	4.23
26	2	5	2	2.46	1.54	1.11	4.52
28	5	5	4	1.33	2.25	0.91	4.40
37	4	4	1	1.16	1.70	0.50	4.98
21	5	1	1	1.39	2.94	0.36	6.23
14	2	4	3	1.09	2.56	0.14	6.92
10	5	4	1	1.99	2.16	0.13	6.68
9	1	2	5	1.61	1.63	0.06	8.95
12	2	5	1	1.16	1.25	0.04	9.31

TABLE VII

COMPARISON OF IMAGE CLASSIFICATION PERFORMANCE FOR TARGET NETWORK BLOCKS PRODUCED THROUGH THE FIRST BLOCKS-LEVEL SEARCH AND THE SECOND BLOCKS-LEVEL SEARCH

Blocks-level search method	#Para (M)	Err. (%)
First blocks-level search	**0.64**	**3.19**
Second blocks-level search	0.65	3.39

possible to directly search for the block topology on the architecture of the target network to obtain a higher performance architecture. Therefore, we choose the individual network-level architecture parameters in the fifth row of Table III(a) to construct the proxy network of the search block topology and conduct experiments on CIFAR10. The cost of this experiment is extremely high. The experiment needs to search on 96G GPU memory, and the obtained block architecture constructs the target network according to the individual network-level architecture parameters in the fifth row of Table III(a). The target network is based on the same training pipeline in Table III, and the results obtained without data augmentation (CutOut) are shown in the second row of Table VII. It can be seen from Table VII that the target network obtained from the second search is weaker in performance and obtains a larger size of parameters. This may be because the proxy network for the second search block is too complicated, which leads to the block topology not being well-optimized.

V. CONCLUSION

In this article, we present EG-DRATS, a multiobjective evolutionary algorithm for neural architecture searching. EG-DRATS has completely designed the network architecture at two stages through block-level search and network-level search. We develop the gradient-based relaxation method for designing blocks with high performance and low complexity by the enhanced gradient. Furthermore, the proposed framework applies the evolutionary multiobjective algorithm to complete the automatic design from blocks to the target network. This method provides a reliable technique to optimize neural networks to achieve high-performance deployment on resource-constrained devices. The algorithm is validated on CIFAR10 and CIFAR100. What is more, at the network-level search stage, we analyze the relationships among network parameters, FLOPS, and inference time. We find that the inference time is directly related to the network depth. Optimizing the network architecture from the three objectives of network parameter size, classification performance, and inference time is our future research direction.

REFERENCES

[1] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," *Commun. ACM*, vol. 60, no. 2, pp. 84–90, Jun. 2012, doi: 10.1145/3065386.

[2] C. Szegedy et al., "Going deeper with convolutions," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2015, pp. 1–9, doi: 10.1109/CVPR.2015.7298594.

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

[4] R. Shang, J. He, J. Wang, K. Xu, L. Jiao, and R. Stolkin, "Dense connection and depthwise separable convolution based CNN for polarimetric SAR image classification," *Knowl.-Based Syst.*, vol. 194, Apr. 2020, Art. no. 105542.

[5] X. Zhang, X. Zhou, M. Lin, and J. Sun, "ShuffleNet: An extremely efficient convolutional neural network for mobile devices," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 6848–6856, doi: 10.1109/CVPR.2018.00716.

[6] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L. Chen, "MobileNetV2: Inverted residuals and linear bottlenecks," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 4510–4520, doi: 10.1109/CVPR.2018.00474.

[7] F. E. Fernandes and G. G. Yen, "Automatic searching and pruning of deep neural networks for medical imaging diagnostic," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 32, no. 12, pp. 5664–5674, Dec. 2021, doi: 10.1109/TNNLS.2020.3027308.

[8] W. Hu, J. Jin, T.-Y. Liu, and C. Zhang, "Automatically design convolutional neural networks by optimization with submodularity and supermodularity," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 31, no. 9, pp. 3215–3229, Sep. 2020, doi: 10.1109/TNNLS.2019.2939157.

[9] Y. Sun, B. Xue, M. Zhang, and G. G. Yen, "Completely automated CNN architecture design based on blocks," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 31, no. 4, pp. 1242–1254, Apr. 2020, doi: 10.1109/TNNLS.2019.2919608.

[10] T. Elsken, J. H. Metzen, and F. Hutter, "Neural architecture search: A survey," *J. Mach. Learn. Res.*, vol. 20, no. 55, pp. 1–21, 2019.

[11] Z. Lu et al., "Multi-objective evolutionary design of deep convolutional neural networks for image classification," *IEEE Trans. Evol. Comput.*, vol. 25, no. 2, pp. 277–291, Apr. 2021, doi: 10.1109/TEVC.2020.3024708.

[12] Z. Zhong et al., "BlockQNN: Efficient block-wise neural network architecture generation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 43, no. 7, pp. 2314–2328, Jul. 2021, doi: 10.1109/TPAMI.2020.2969193.

[13] M. Tan, R. Pang, and Q. V. Le, "EfficientDet: Scalable and efficient object detection," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2020, pp. 10781–10790.

[14] O. Russakovsky et al., "ImageNet large scale visual recognition challenge," *Int. J. Comput. Vis.*, vol. 115, no. 3, pp. 211–252, Dec. 2015, doi: 10.1007/s11263-015-0816-y.

[15] X. Wang, Y. Peng, L. Lu, Z. Lu, M. Bagheri, and R. M. Summers, "ChestX-ray8: Hospital-scale chest X-ray database and benchmarks on weakly-supervised classification and localization of common thorax diseases," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 2097–2106.

[16] H. Zhang, K. Hao, L. Gao, B. Wei, and X. Tang, "Optimizing deep neural networks through neuroevolution with stochastic gradient descent," *IEEE Trans. Cognit. Develop. Syst.*, early access, Jan. 26, 2022, doi: 10.1109/TCDS.2022.3146327.

[17] Y. Liu, Y. Sun, B. Xue, M. Zhang, G. G. Yen, and K. C. Tan, "A survey on evolutionary neural architecture search," *IEEE Trans. Neural Netw. Learn. Syst.*, early access, Aug. 6, 2021, doi: 10.1109/TNNLS.2021.3100554.

[18] 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, doi: 10.1109/TCYB.2020.2983860.

[19] K. O. Stanley and R. Miikkulainen, "Evolving neural networks through augmenting topologies," *Evol. Comput.*, vol. 10, no. 2, pp. 99–127, 2002, doi: 10.1162/106365602320169811.

[20] Y. Sun, H. Wang, B. Xue, Y. Jin, G. G. Yen, and M. Zhang, "Surrogate-assisted evolutionary deep learning using an end-to-end random forest-based performance predictor," *IEEE Trans. Evol. Comput.*, vol. 24, no. 2, pp. 350–364, Apr. 2020.

[21] V. Mnih et al., "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, 2015, doi: 10.1038/nature14236.

[22] B. Colson, P. Marcotte, and G. Savard, "An overview of bilevel optimization," *Ann. Oper. Res.*, vol. 153, no. 1, pp. 235–256, Sep. 2007, doi: 10.1007/s10479-007-0176-2.

[23] H. Liu, K. Simonyan, and Y. Yang, "DARTS: Differentiable architecture search," in *Proc. Int. Conf. Learn. Represent.*, 2018, pp. 1–13.

[24] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: NSGA-II," *IEEE Trans. Evol. Comput.*, vol. 6, no. 2, pp. 182–197, Apr. 2002, doi: 10.1109/4235.996017.

[25] 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. (ICCV)*, Oct. 2019, pp. 1294–1303, doi: 10.1109/ICCV.2019.00138.

[26] J. H. Holland, *Adaptation in Natural and Artificial Systems: An Introductory Analysis With Applications to Biology, Control, and Artificial Intelligence*. Cambridge, MA, USA: MIT Press, 1992.

[27] L. Xie and A. Yuille, "Genetic CNN," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Oct. 2017, pp. 1388–1397, doi: 10.1109/ICCV.2017.154.

[28] B. Zoph and Q. V. Le, "Neural architecture search with reinforcement learning," 2017, *arXiv:1611.01578*.

[29] B. Zoph, V. Vasudevan, J. Shlens, and Q. V. Le, "Learning transferable architectures for scalable image recognition," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 8697–8710, doi: 10.1109/CVPR.2018.00907.

[30] Z. Zhong, J. Yan, W. Wu, J. Shao, and C.-L. Liu, "Practical block-wise neural network architecture generation," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 2423–2432.

[31] C. Liu et al., "Progressive neural architecture search," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*. Cham, Switzerland: Springer, 2018, pp. 19–35.

[32] H. Dong, B. Zou, L. Zhang, and S. Zhang, "Automatic design of CNNs via differentiable neural architecture search for PolSAR image classification," *IEEE Trans. Geosci. Remote Sens.*, vol. 58, no. 9, pp. 6362–6375, Sep. 2020, doi: 10.1109/TGRS.2020.2976694.

[33] X. Zhang, Z. Huang, N. Wang, S. Xiang, and C. Pan, "You only search once: Single shot neural architecture search via direct sparse optimization," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 43, no. 9, pp. 2891–2904, Sep. 2021, doi: 10.1109/TPAMI.2020.3020300.

[34] Y. Hua, Y. Jin, and K. Hao, "A clustering-based adaptive evolutionary algorithm for multiobjective optimization with irregular Pareto fronts," *IEEE Trans. Cybern.*, vol. 49, no. 7, pp. 2758–2770, Jul. 2019.

[35] Y. Jin and B. Sendhoff, "Pareto-based multiobjective machine learning: An overview and case studies," *IEEE Trans. Syst., Man, Cybern. C, Appl. Rev.*, vol. 38, no. 3, pp. 397–415, May 2008, doi: 10.1109/TSMCC.2008.919172.

[36] Y.-H. Kim, B. Reddy, S. Yun, and C. Seo, "Nemo: Neuro-evolution with multiobjective optimization of deep neural network for speed and accuracy," in *Proc. ICML*, 2017, pp. 1–8.

[37] J.-D. Dong, A.-C. Cheng, D.-C. Juan, W. Wei, and M. Sun, "Dpp-Net: Device-aware progressive search for pareto-optimal neural architectures," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, 2018, pp. 517–531.

[38] J. Huang, W. Sun, and L. Huang, "Deep neural networks compression learning based on multiobjective evolutionary algorithms," *Neurocomputing*, vol. 378, pp. 260–269, Feb. 2020.

[39] J. Jiang, F. Han, Q. Ling, J. Wang, T. Li, and H. Han, "Efficient network architecture search via multiobjective particle swarm optimization based on decomposition," *Neural Netw.*, vol. 123, pp. 305–316, Mar. 2020.

[40] M. Suganuma, M. Kobayashi, S. Shirakawa, and T. Nagao, "Evolution of deep convolutional neural networks using Cartesian genetic programming," *Evol. Comput.*, vol. 28, no. 1, pp. 141–163, Mar. 2020, doi: 10.1162/evco_a_00253.

[41] L. Bottou, "Large-scale machine learning with stochastic gradient descent," in *Proc. COMPSTAT*. Cham, Switzerland: Springer, 2010, pp. 177–186.

[42] T. DeVries and G. W. Taylor, "Improved regularization of convolutional neural networks with cutout," 2017, *arXiv:1708.04552*.

[43] E. Real, A. Aggarwal, Y. Huang, and Q. V. Le, "Regularized evolution for image classifier architecture search," in *Proc. AAAI Conf. Artif. Intell.*, vol. 33, 2019, pp. 4780–4789.

[44] K. S. H. Liu, O. Vinyals, C. Fernando, and K. Kavukcuoglu, "Hierarchical representations for efficient architecture search," in *Proc. ICLR*, 2018, pp. 1–13.

[45] M. Suganuma, S. Shirakawa, and T. Nagao, "A genetic programming approach to designing convolutional neural network architectures," in *Proc. Genetic Evol. Comput. Conf.*, Berlin, Germany, Jul. 2017, pp. 497–504.

[46] H. Pham, M. Guan, B. Zoph, Q. Le, and J. Dean, "Efficient neural architecture search via parameters sharing," in *Proc. 35th Int. Conf. Mach. Learn.*, 2018, pp. 4095–4104.

[47] H. Zhang, Y. Jin, R. Cheng, and K. Hao, "Efficient evolutionary search of attention convolutional networks via sampled training and node inheritance," *IEEE Trans. Evol. Comput.*, vol. 25, no. 2, pp. 371–385, Apr. 2021.

[48] A. Ashok, N. Rhinehart, F. Beainy, and K. M. Kitani, "N2N learning: Network to network compression via policy gradient reinforcement learning," in *Proc. Int. Conf. Learn. Represent.*, 2018, pp. 1–20.

[49] Y. He, J. Lin, Z. Liu, H. Wang, L.-J. Li, and S. Han, "AMC: Automl for model compression and acceleration on mobile devices," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, 2018, pp. 784–800.

[50] S. Cao, X. Wang, and K. M. Kitani, "Learnable embedding space for efficient neural architecture compression," in *Proc. Int. Conf. Learn. Represent.*, 2018, pp. 1–17.

[51] X. Li, S. Li, B. Omar, F. Wu, and X. Li, "ResKD: Residual-guided knowledge distillation," *IEEE Trans. Image Process.*, vol. 30, pp. 4735–4746, 2021.

[52] T.-B. Xu and C.-L. Liu, "Deep neural network self-distillation exploiting data representation invariance," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 33, no. 1, pp. 257–269, Jan. 2022.

[53] M. Loni, S. Sinaei, A. Zoljodi, M. Daneshtalab, and M. Sjödin, "DeepMaker: A multi-objective optimization framework for deep neural networks in embedded systems," *Microprocessors Microsyst.*, vol. 73, Mar. 2020, Art. no. 102989.

[54] H. Cai, L. Zhu, and S. Han, "ProxylessNAS: Direct neural architecture search on target task and hardware," in *Proc. Int. Conf. Learn. Represent.*, 2018, pp. 1–13.

[55] T. Elsken, J. H. Metzen, and F. Hutter, "Efficient multi-objective neural architecture search via Lamarckian evolution," in *Proc. Int. Conf. Learn. Represent.*, 2018, pp. 1–23.

[56] Y. Xue, P. Jiang, F. Neri, and J. Liang, "A multi-objective evolutionary approach based on graph-in-graph for neural architecture search of convolutional neural networks," *Int. J. Neural Syst.*, vol. 31, no. 9, Sep. 2021, Art. no. 2150035.

[57] S. Kim, H. Kwon, E. Kwon, Y. Choi, T. H. Oh, and S. Kang, "MDARTS: Multi-objective differentiable neural architecture search," in *Proc. Design, Autom. Test Eur. Conf. Exhib. (DATE)*, Feb. 2021, pp. 1344–1349, doi: 10.23919/DATE51398.2021.9474068.

[58] Z. Lu et al., "NSGA-Net: Neural architecture search using multi-objective genetic algorithm," in *Proc. Genetic Evol. Comput. Conf.*, Jul. 2019, pp. 419–427.

[59] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *IEEE CVPR*, Jun. 2017, pp. 4700–4708.

[60] Q. Wang, B. Wu, P. Zhu, P. Li, W. Zuo, and Q. Hu, "ECA-Net: Efficient channel attention for deep convolutional neural networks," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2020, pp. 11531–11539, doi: 10.1109/CVPR42600.2020.01155.

Haichao Zhang received the M.S. degree in textile engineering from the Inner Mongolia University of Technology, Hohhot, China, in 2016. He is currently pursuing the Ph.D. degree with the College of Information Science and Technology, Donghua University, Shanghai, China.

His research interests include deep learning, machine vision, image processing, and artificial intelligence.

Kuangrong Hao received the B.S. and M.S. degrees in mechanical engineering from the Hebei University of Technology, Tianjin, China, in 1984 and 1989, respectively, the M.S. degree in applied mathematics from the École normale supérieure Paris-Saclay, Gif-sur-Yvette, France, in 1991, and the Ph.D. degree in computer sciences from Ecole Nationale des Ponts et Chaussées, Paris, France, in 1995.

She is currently a Full Professor with the College of Information Science and Technology, Donghua University, Shanghai, China. She has authored more than 200 technical articles and five research monographs. Her research interests include intelligent perception, intelligent systems, network intelligence, robot control, and intelligent optimization of textile industrial process.

Lei Gao (Member, IEEE) received the B.S. and Ph.D. degrees in electrical engineering from Donghua University, Shanghai, China, in 2001 and 2006, respectively.

He is currently a Principal Research Scientist with Australia's Commonwealth Scientific and Industrial Research Organization (CSIRO), Urrbrae, SA, Australia, and also an external Ph.D. Supervisor with The University of Western Australia, Australia, and Deakin University, Victoria, Australia. He has authored over 130 papers, mostly in prestigious journals, such as Nature, Nature Water, and Nature Communications. His research interests include complex system modeling and optimization, big data modeling, machine learning, robust decision-making, and computational sustainability.

Dr. Gao was a recipient of the Early Career Research Excellence Prize of the Modelling and Simulation Society of Australia and New Zealand, CSIRO Chairman's Medal for Science Excellence, and Julius Career Award (for exceptional early to mid-career scientists in CSIRO).

Xue-song Tang received the Ph.D. degree in computer science from Fudan University, Shanghai, China, in 2015.

He is currently a Lecturer with the College of Information Science and Technology, Donghua University, Shanghai. His research interests include deep learning, machine vision, image processing, interdiscipline research on brain science, and artificial intelligence.

Bing Wei (Member, IEEE) received the Ph.D. degree from the College of Information Science and Technology, Donghua University, Shanghai, China, in 2020.

He was a Joint Ph.D. Student with the Department of Electrical, Computer, and Biomedical Engineering, University of Rhode Island, Kingston, RI, USA, from 2018 to 2019. He is currently a Lecturer with the College of Information Science and Technology, Donghua University. His research interests include biological computing, deep learning, image processing, and artificial intelligence.