2021 6th International Conference on Computational Intelligence and Applications (ICCIA) | 978-1-6654-3933-6/21/$31.00 ©2021 IEEE | DOI: 10.1109/ICCIA52886.2021.00061

ME-DARTS: Introduce Multi-stage Evolution to Improve Differentiable Architecture Search

Yusu Chen
College of Electronics and Information Engineering
Shenzhen University
Shenzhen, China
1810262047@email.szu.edu.cn

Rui Shi
College of Electronics and Information Engineering
Shenzhen University
Shenzhen, China
1800261012@email.szu.edu.cn

Jianping Luo*
College of Electronics and Information Engineering
Shenzhen University
Shenzhen, China
ljp@szu.edu.cn

***Abstract*—Neural Network Architecture Search (NAS) can automatically learn to obtain deep neural networks for specific scenarios. Differentiable Architecture Search (DARTS) makes the NAS process more efficient. However, its performance is limited by the tendency of DARTS to fall into architectural local optima. In this study, we introduce evolutionary algorithm (EA) into DARTS and propose multi-stage evolution to improve the performance of DARTS, named ME-DARTS. We construct a population containing multiple networks and divide the search process into three stages. An inadequate-to-adequate search strategy and datasets approximation computation are used to accelerate the search process. The networks in the population will be promoted by the evolutionary operators, and finally a network architecture with good performance will be selected from the population. Our algorithm achieves a test error of 2.6% on CIFAR10 and 15.68% on CIFAR100, demonstrating the effectiveness of ME-DARTS.**

Keywords—deep neural network, evolutionary algorithm, network architecture search, DARTS

I. INTRODUCTION

Automatic machine learning (Auto-ML) has received a lot of attention in recent years. The development of neural network architecture search (NAS) has brought the design of neural networks to the stage of automation. The neural network obtained through NAS can be applied to a variety of tasks, such as recommendation system [1], image recognition [2], [3], image classification [4]-[9] and semantic segmentation [10], etc. Compared with manually designed neural networks such as VGGNet [11] and ResNet [12], the neural network through architecture search can also achieve good performance.

The current main methods of NAS include evolutionary algorithm (EA-based) [1], [8], [13], reinforcement learning (RL-based) [7], [9], [14], [15] and methods based on gradient descent (Gradient-based) [4], [5], [16]. These methods firstly construct a search space, and then use specific methods to search in the search space to obtain the optimal network architecture. Among them, most NAS based on evolutionary algorithms or reinforcement learning need to consume a lot of time and computing resources, such as Google's large-scale computing [13] and NasNet [9], etc. Liu et al [5] proposed the Differentiated Architecture Search (DARTS) to continuous the search space, and then applied gradient descent to the search process. The contribution of DARTS is to make the process of NAS more efficient. Compared with EA-based and RL-based algorithms, DARTS only needs a few GPU-days to obtain great results.

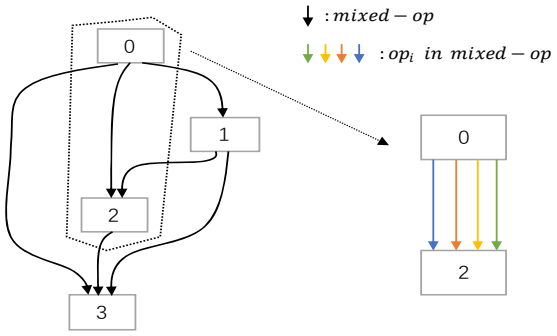

Fig. 1. A cell structure of networks in DARTS

In DARTS, a cell structure in a hyper-network consists of computing nodes that are connected by mixed operation (mixed-op). As shown in Fig. 1, mixed-op contains multiple neural network operations op_i, and DARTS assigns every op_i an architecture parameter α_i. With the $softmax(\cdot)$ function, we use $softmax(\alpha_i)$ as the weights of op_i in the mixed-op, and then architecture parameters α_i is updated through the gradient descent method. The disadvantage of GL-based is that it is easy

978-1-6654-3933-6/21/$31.00 ©2021 IEEE
DOI 10.1109/ICCIA52886.2021.00061

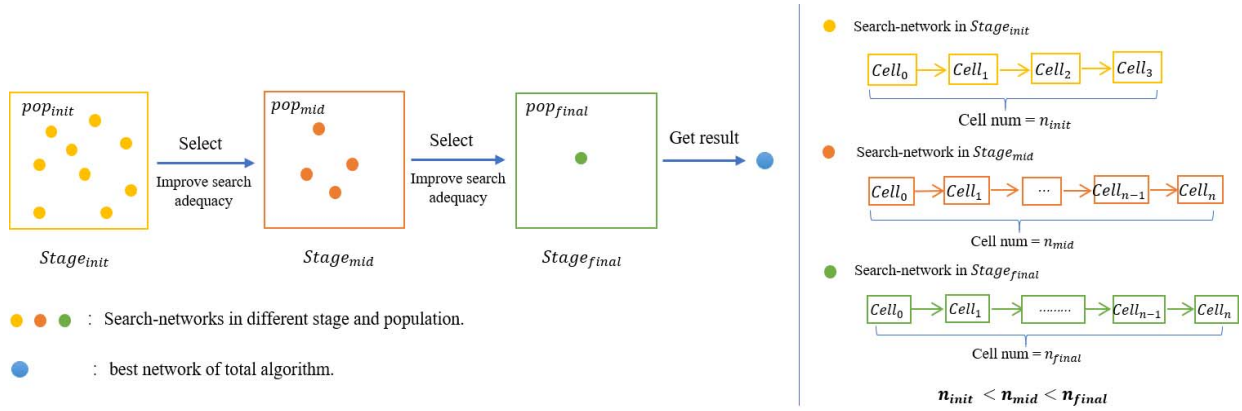

Fig. 2. Schematic diagram of main algorithms in ME-DARTS

to fall into local optimal. The value of α_i is easy to fall into a local optimal state and the final searched structure may only be a local optimal solution. Evolutionary algorithm (EA) selects the better performing individuals from the population through a series of operators. In the above process, the variability among individuals is always maintained, and it is not easy to fall into local optimum.

In this work, we combine the EA with DARTS and propose an improved algorithm based on population and evolutionary computation. To retain the advantages of the evolutionary algorithm and to ensure the efficiency of the computation, we made some improvements. If all individuals in the population directly use the original DARTS setup to search the network architecture, the computational cost must be expensive. Depending on the adequacy of the search process, we divide the whole process into three stages: $stage_{init}$, $stage_{mid}$ and $stage_{final}$. The population consists of search networks, and the size of the population decreases as the stage progresses, while the depth of the network individuals increases.

As shown in Fig. 2, the algorithm is named ME-DARTS (Multi-stage Evolution to improve DARTS). To improve computational efficiency, approximate datasets are created for searching. The approximate dataset consists of a partial dataset obtained by sampling from the original dataset (D_{sample}) and a low-resolution dataset obtained by scaling the images (D_{low}). For the selection process of network individuals in the population, we use non-dominated sorting and Pareto optimal front [17]. In order to retain the dominant individuals and ensure the diversity of the population, evolution-related operations are used in ME-DARTS, such as inheritance and crossover.

ME-DARTS used CIFAR10 and CIFAR100 datasets to search for the high-performance network architectures and evaluated the network on CIFAR10 and CIFAR100. After searching on single RTX 2080Ti, ME-DARTS achieved a test error of 2.6% on the CIFAR10 with 0.55 GPU days and a test error of 15.84% on the CIFAR100 with 0.45 GPU days. Compared to DARTS, the search consumption of our algorithm is greatly reduced. The work related to DARTS includes P-DARTS [4], PC-DARTS [18], etc. Unlike these approaches, the main contributions and innovation in our work are followings:

- We combine EA and DARTS to make the search process of network architecture not easily fall into a local optimum, while adding individual inheritance, crossover and other operations to obtain more diverse network architectures with the diversity of populations.
- Constructing approximate datasets for computational acceleration. Both non-dominated ranking and Pareto frontier selection methods are used to select the network architecture in the population.
- We propose a progressive search strategy and gradually refines the search conditions in each stage to allocate computational resources, thus gradually approaching the final network architecture with good performance.

II. Related Work

Neural network architecture search (NAS) is advancing very rapidly. EA-based and RL-based NAS methods are dominating for a long time, and these methods have achieved good performance results, such as large-scale evolution [13] and AmoebaNet [8] in EA-based. But these methods are computationally expensive, large-scale evolution takes 11 days on 250 GPUs and the latter spends 7 days on 450 GPUs on CIFAR10. EIGEN [1], which is also EA-based, although the highest accuracy is not as good as the above algorithms, has a more efficient algorithm that requires only 2 GPU days to obtain a good result. Among the RL-based methods, NasNet [9] is a representative work that uses a meta-controller to guide the generation of the network architecture with a time consumption of 1800 GPU days for the search phase. In addition to EA-based and RL-based methods, ENAS [7] applies weight sharing to the sub-architecture of the network and the training time consumption of the network is saved to 1 GPU day, while PNAS [10] and E2EPP [19] are based on the construction of an agent model for network architecture search, and the time consumption of the former is 350 GPU days, while the latter only costs 8.5 GPU Days.

In addition to the NAS methods above, DARTS [5] proposes a method to serialize the neural network architecture and use gradient descent for network architecture search, and after 1.5 GPU Days of searching on a single GTX1080ti, DARTS

obtained a state-of-the-art result at that time on several datasets such as CIFAR10 and CIFAR100, which was a great efficiency improvement compared with methods such as EA-based NAS and RL-based NAS. On the basis of DARTS, many excellent improvement schemes have been derived. For example, P-DARTS [4] proposes a progressive DARTS method that solves the direct gap problem between the search network and the evaluation network, and it makes the search process more efficient. ProxylessNAS [16] abandoned the approach of using traditional proxy frameworks or proxy datasets, searched directly on the target task. There are also DARTS+ [20], FairDARTS [21] and so on, all these works are optimized to Improve the shortcomings of DARTS performance and reduce search time consumption.

III. METHOD

A. Preliminary: DARTS

In DARTS, a CNN or RNN network architecture consists of multiple cells, and DARTS is to find the appropriate cell. According to the function of cell, the cell of the network in DARTS can be divided into normal cell and reduction cell. The size of the image remains unchanged after input to the normal cell, while the size decreases after input to the reduction cell. Cell is a directed acyclic graph (DAG) containing N nodes, each of which is connected to the node before it. Each node in the Cell represents an implicit feature value, and the connected line, represents the flow of information between nodes and nodes, contains multiple operations including convolution, pooling, skip-connect, zero, and so on. Each Cell has two inputs and one output. For CNN structures, the current cell C_k^{CNN} receive the output from the two previous cells C_{k-2}^{CNN} and C_{k-1}^{CNN}, while for RNNs, the input of the current cell C_k^{RNN} is the output of the current cell C_k^{RNN} and the output of the previous cell C_{k-1}^{RNN}. Assume that the entire search space is O, a cell has N nodes $\{x_0, x_1, \dots, x_{N-1}\}$, $e_{i,j}$ denotes a line from node x_i to node x_j, $o_{i,j}^m(\cdot)$ is an operation on this concatenation and $o(\cdot) \in O$, Each operation has a coefficient $\alpha_{i,j}^{o^m}$, All the operation coefficients from node x_i to node x_j can be written as vectors $(\alpha_{i,j}^{o^1}, \alpha_{i,j}^{o^2}, \dots \alpha_{i,j}^{o^L})$, Then the output value of the node x_j in cell can be written as:

$$x_j = \sum_{i<j} o_{i,j}(x_i) \tag{1}$$

The formula above indicates that for each intermediate node x_j, its output is the concatenates of the outputs of all predecessor nodes x_i on the channel dimension. DARTS uses the $softmax(\cdot)$ function to transform the architecture parameter $\alpha_{i,j}^o$ corresponding to $o(x)$ as the weight of $o(x)$, so $o_{i,j}(x)$ is the weighted sum of all operations from node x_i to node x_j, it formulated as (2):

$$\bar{o}_{i,j}(x) = \sum_{o \in O} \frac{exp(\alpha_{i,j}^o)}{\sum_{o' \in O} exp(\alpha_{i,j}^{o'})} o(x) \tag{2}$$

The above is the mathematical expression of a hyper-network, for which w is the weight parameter of the network and α is the architecture parameter. L_{train} and L_{val} are the training loss and validation loss of DARTS in the search process, then the whole training process of DARTS is doing the bi-leveled optimization problem shown below:

$$\min_{\alpha} \; L_{val}(w^*(\alpha), \alpha) \tag{3}$$

$$s.t. \; w^*(\alpha) = arg \min_{w} \; L_{train}(w, \alpha) \tag{4}$$

After the search stage, for the connection $e_{i,j}$ from node x_i to node x_j, the discretization operation is shown as (5):

$$o_{i,j} = arg \max_{o \in O} \frac{exp(\alpha_{i,j}^o)}{\sum_{o' \in O} exp(\alpha_{i,j}^{o'})} \tag{5}$$

thus, the specific operation from node x_i to node x_j satisfying $i < j$ is obtained. For node x_j, $\max_{o \in O} \frac{exp(\alpha_{i,j}^o)}{\sum_{o' \in O} exp(\alpha_{i,j}^{o'})}$ is used as a pick indicator to select the two largest connections as input among the remaining connections mentioned above. For more technical details, please refer to the original DARTS paper [5].

B. Limitations of DARTS

DARTS is a gradient-based NAS method that updates the network weight parameters w_i and the architecture parameters α_i by combined optimization during the search process, and finally discretizes the parameter α_i using $argmax(\cdot)$ to obtain a final network architecture. Although DARTS is efficient and has good performance compared to other NAS methods, it suffers from the problem of falling into local optimum. In the search process of DARTS, the algorithm will tend to choose operation of skip-connect, resulting in an increase in the number of skip-connect operations among all operations. This phenomenon has been mentioned in P-DARTS, DARTS+ and other DARTS-related work [4][5][21], for the following reasons:

- The skip-connect operation is friendly for the process of gradient descent because it means that gradient descent has a faster path, and then the search process tends to increase the value of the architecture parameter corresponding to skip-connect, leading to more skip-connect operations when discretizing the hyper-network.
- In the search process, w_i and α_i are both cooperative and competitive. At the beginning, these two variables are cooperative, when the epoch reaches a certain number, the two change from cooperation to a competition, and w_i is more dominant. The performance of the whole network in will become worse. Moreover, the architectural parameter α_i of other operations in hyper-network cannot compete with the architectural parameter α_{sc} of skip-connect operations, which limits the performance of the network.

TABLE I. STAGED PROGRESSIVE SEARCH STRATEGY

Algorithm: Staged progressive search strategy

When in $stage_{init}$

Initialize network G_{init}^{i} and then get population pop_{init} which population size is pop_size_{init}.

$\Theta \leftarrow SetConfigurations(stage_{init})$

For G_{init}^{i} in pop_{init}

$G_{mid}^{i} \leftarrow searchTrain(G_{init}^{i}, \Theta)$

When in $stage_{mid}$

Select network G_{mid}^{i} From pop_{init}, and get population pop_{mid}.

$\Theta \leftarrow SetConfigurations(stage_{mid})$

For G_{mid}^{i} in pop_{mid}

$G_{final}^{i} \leftarrow searchTrain(G_{mid}^{i}, \Theta)$

When in $stage_{final}$

Select single network G_{final} From pop_{mid}

$\Theta \leftarrow SetConfigurations(stage_{final})$

$G_{final} \leftarrow searchTrain(G_{final}, \Theta)$

Return G_{final}

C. Introducing the Evolutionary Framework into DARTS

DARTS uses the one-shot [16], [22], [23] idea in constructing the network, i.e., fusing multiple operations together to form a hypernetwork, and then sampling from the hypernetwork according to a specific algorithm to obtain the network architecture. EA constructs a population with a wide range of initial individuals, and diversifies the solution of the network architecture through a series of evolutionary operators. In ME-DARTS, each network individual is searched through the method of DARTS. We first construct an initial population pop_{init}, which consists of hyper-networks G_i with the same search space and connection ways as the hyper-networks $\hat{G}$ in DARTS, except that G_i in pop_{init} has fewer cells and less channels than $\hat{G}$. During the search training, we do not restrict the random seeds in order to initialize each individual G_i of pop_{init} randomly and avoid the convergence among G_i. So, these hyper-networks can obtain more diverse architectures after search training. Then the network individuals with better performance in this initial population are selected according to the selection strategy of Pareto optimal front, and these individuals are enhanced with an evolutionary strategy to gradually improve the performance. Obviously, the construction of multiple populations of hyper-network individuals will increase the computational consumption. To solve this problem, we propose a progressive search strategy based on EA and an approximate training method that can optimize the learning cost while ensuring the effectiveness.

1) Progressive search strategy: If each network individual in the initial population pop_{init} in ME-DARTS is fully searched, this will bring unbearable computational resource consumption. To ensure both the quality of search training and non-expensive computational con-sumption, the search process in the algorithm is divided into three phases: $stage_{init}$, $stage_{mid}$ and $stage_{final}$. With the progression of the search stage, the adequacy of the search gradually increases. The adequacy of the search is distinguished by the different settings of hyper-parameters such as number of cells (num_{cell}), number of channels ($num_{channel}$), epoch and other hyperparameters of the search network. The flow of this progressive search strategy algorithm is shown in TABLE I.

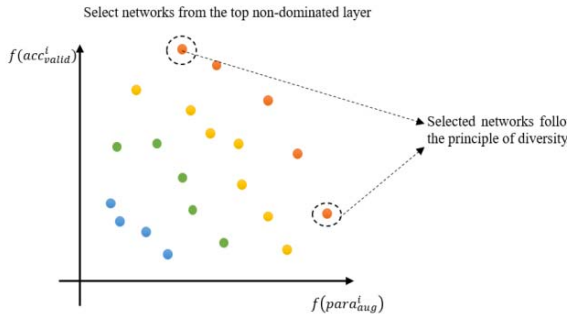

Fig. 3. Non-dominated sorting and Pareto frontier selection

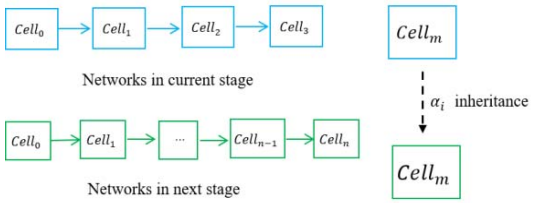

Fig. 4. Inheritance of network architecture parameters

In $stage_{init}$, we construct an initial population pop_{init}, in which contains multiple initial search networks, and ps_{init} indicates the number of networks in the population. We cut the num_{cell}, $num_{channel}$ of search network and searching epoch in $stage_{init}$ to some extent, and then selected the better-performing network individuals from this initial population after simply trained. We call these search networks with reduced hyper-parameters simple search networks. In fact, constructing a population of simple search networks and simply searching them cannot accurately obtain the actual performance of these network individuals, but in this search process, we only focus on the relative performance and the rankings of network individuals in the population. The top-ranked individuals will be selected to form the population pop_{mid}, and the algorithm will proceed to the $stage_{mid}$ while doing the followings:

- Improve the adequacy of the search, such as increasing num_{cell}, the number of search epoch, etc.
- Performing a new search process for network individuals in the new population.

After the above two works, the well performed individuals from the population pop_{mid} will be selected to form a new population pop_{final}, and the algorithm will proceed to $stage_{final}$. The individuals of pop_{final} continue the two works to obtain the final network individual G_{final}.

Progressive multi-stage search strategy has a certain rationality. Superior network individuals selected in current stage will get more computational resources in the next stage. Then after multiple rounds of selection, the network individuals with better performance can be obtained after $stage_{final}$.

2) Dataset approximation calculation: We propose the method of dataset approximation calculation in order to obtain the relative ranking of network individuals while minimizing the computational resource consumption in the search process. Dataset approximation calculation includes two steps: constructing an approximate dataset and using the approximate dataset for search. The approximate dataset consists of a partial sample (D_{sample}) dataset and a low-resolution dataset (D_{low}).

The purpose of constructing D_{sample} is to reduce the computational consumption by reducing the number of training samples, while keeping the performance at the same level as the full-sample dataset training. When constructing a partial sample dataset for classification, such as CIFAR10, we need to ensure that the number of images in each category is equal so that the data distribution of the partial sample dataset is close to that of the full sample dataset.

The low-resolution dataset D_{low} refers to the processing of the input dataset using the image scaling sampling method, after obtaining a set of low-resolution images, and then the low-resolution images are input into the search network. For example, for the CIFAR10 dataset, the size of each image is $H_s \times W_s \times C = 32 \times 32 \times 3$. In this work, the LANCZOS method is used to scale the images to the size of $H_l \times W_l \times C = 16 \times 16 \times 3$, and the width and height are reduced to half of the original size. These low-resolution images are used to search for network individuals, and standard-resolution images are used for performance evaluation.

The approximate datasets D_{sample} and D_{low} are only used for the search process, while the evaluation process still uses the standard dataset. As the search phase progresses, the number of data samples in D_{sample} increases, and D_{low} maintains half of the resolution scaling.

3) Non-dominated sorting and Pareto frontier selection: In $stage_{init}$ of the progressive search strategy, the individuals with better performance will be selected from pop_{init} to form a population pop_{mid} of the later progress $stage_{mid}$. In this work, we follow the idea of NSGA-II [17] to perform non-dominated sorting and Pareto frontier selection of network individuals in the population pop_{init} of $stage_{init}$, and turn the selection of network individuals into multi-objective optimization problem. For each search network G_i, the two key metrics to measure the network are the validation accuracy (acc^i_{valid}) and the number of parameters ($para^i_{aug}$) in the search process. $para^i_{aug}$ is the number of parameters of evaluation network $\hat{G}_i$. $\hat{G}_i$ has 20 cells, and the operations in each cell are discretized and sampled according to architecture parameters α_i from the mixed operations.

For each network individual $G_i(acc^i_{valid}, para^i_{aug})$, we consider that the larger the targets $f(acc^i_{valid})$ and $f(para^i_{aug})$ represent the better network performance. As shown in (6), the function $f(\cdot)$ is to normalize this variable. acc^i_{valid} represents the current performance of the search network, and $para^i_{aug}$ represents the potential performance of the network. The larger $param_{aug}$ means the higher complexity of the network and more convolutional operations, so that more information can be extracted. When the algorithm proceeds to the next stage, the adequacy of the search is improved accordingly, and the network with larger $param_{aug}$ can be trained more adequately to show the potential performance.

$$f(x_i) = \frac{x_i}{x_{max} - x_{min}}, i \in \{1, \dots, n\} \quad (6)$$

The above network individual selection problem is a maximization multi-objective problem, and each network individual G_i in the population may not satisfy that both $f(acc^i_{valid})$ and $f(para^i_{aug})$ are maximum at the meantime. we introduce non-dominated sorting to sort network individuals in the population and obtain non-dominated solutions. Pareto frontier selection is used to select suitable networks from the non-dominated solutions. Both methods follow the principle of population diversity. The principle of diversity of objective function first means that the selected network individuals should have some differences in the performance of (acc^i_{valid}) and $f(para^i_{aug})$ so that the network individuals will not converge.

$$if \ \forall i \in \{1, \dots, n\}, f_i(X_a) < f_i(X_b), then \ X_b \ dominate \ X_a \quad (7)$$

Suppose there exist any two variables X_a and X_b, and a vector $\bar{f}(X) = (f_1(X), f_2(X), \dots, f_n(X))$ consisting of n target components $f_i(i = 1, \dots, n)$, for the maximization problem, the dominance relationship between the variables is shown in (7). The dominance relationship between network individuals is used to perform the non-dominance ranking of network individuals in the population, and the results of the non-dominance ranking are shown in Fig. 3

4) Network inheritance and crossover. In the progression from $stage_{init}$ to $stage_{mid}$, the well performed network individuals $(g_0^{\alpha_0}, g_1^{\alpha_1} \dots, g_m^{\alpha_m})$ are selected from pop_{init}, and then $(\tilde{g}_0^{\alpha_0}, \tilde{g}_1^{\alpha_0} \dots, \tilde{g}_m^{\alpha_m})$ is obtained by $(g_0^{\alpha_0}, g_1^{\alpha_1} \dots, g_m^{\alpha_m})$ after improving the search adequacy to formed pop_{mid}. The architecture parameters α_i of each cell in $(\tilde{g}_0^{\alpha_0}, \tilde{g}_0^{\alpha_0} \dots, \tilde{g}_m^{\alpha_m})$ remain unchanged. The architectural parameter α_i^{mid} of the network individual of pop_{mid} inherits the architectural parameter α_i^{init} of the network individual selected from pop_{init}. Similarly, in the progression from $stage_{mid}$ to $stage_{final}$, there is an inheritance of architectural parameters for the network individuals. As shown in Fig. 4, this process retains the network architecture parameters α_i' from the previous phase and then, in current stage, assigns α_i' to a network containing a larger number of cells and continues the search.

After searching in $stage_{init}$, the architecture of each network is improved, including normal cell and reduction cell. Before $stage_{mid}$, we performed the crossover operation on the previous network individuals to recombine them into new

network individuals so as to improve the diversity of individuals in the pop_{mid}. The process of crossover is to recombine normal cells and reductive cells of different network structures to obtain a new network. Suppose there are network individuals $g_1(\alpha_{nor}^1,\alpha_{red}^1)$ and $g_2(\alpha_{nor}^2,\alpha_{red}^2)$ in the pop_{mid} of $stage_{mid}$, α_{nor} and α_{red} denote the normal cell and the reproduction cell in the individuals respectively. After the crossover of network architecture parameters, new individuals $g_3(\alpha_{nor}^1, \alpha_{red}^2)$ and $g_4(\alpha_{nor}^2, \alpha_{red}^1)$ are obtained. These new network individuals are put into the population pop_{mid} together with the original network individuals and then the start the search of $stage_{mid}$.

IV. Experiments and Results

A. Datasets

CIFAR10 and CIFAR100 datasets are used in MEDARTS for network search and evaluation. Both CIFAR10 and CIFAR100 have 50K RGB images for training and 10K RGB images for testing, each with a fixed spatial resolution of $H_s \times W_s \times C = 32 \times 32 \times 3$. The difference between the two datasets is that there is a total of 10 classes of images in CIFAR10 and a total of 100 classes of images in CIFAR100. In the search training process, we construct partial samples dataset D_{sample}. In accordance with the category sampling method, a portion of images are evenly selected from each category of the original training set, and this portion of images is constructed as a partial sample dataset to ensure that the number of each category in the partial sample dataset is the same. Assuming that the number of partial sample dataset is $nums_{mini}$, then $nums_{mini} < 50000$. We use LANCZOS sampling method to obtain D_{low} with a fixed spatial resolution of $H_l \times W_l \times C = 16 \times 16 \times 3$ before inputting it into the network. When in networks evaluation stage, origin datasets will be used for network evaluation without dataset approximation.

B. Architecture Search

1) Search Setting: In ME-DARTS, the candidate operation space for each Cell is the same as DARTS, with a total of 8 candidate operations, including skip-connect, max-pool-3x3, avg-pool-3x3, sep-conv-3x3, sep-conv-5x5, dil-conv-3x3, dil-conv5x5, zero. As with DARTS, we divide the training set equally into two parts, one for updating the model weights and the other for updating the architecture parameters. The different search settings for each stage in the search process as shown in TABLE II. From $stage_{init}$ to $stage_{final}$, , these search adequacies are gradually improved as shown by the improvement of the number of network cells, epoch and the number of dataset samples, etc.

2) Search Costs: The three stages of the algorithm cost a total of 25 epochs for both the CIFAR10 and CIFAR100 datasets. On the CIFAR10 dataset, ME-DARTS spent 0.55 GPU Days on a RTX 2080Ti for the entire algorithm process.This saves more than 60% of the computation time consumption compared with DARTS. On the CIFAR100 dataset, ME-DARTS spent a total of 0.45 GPU Days.

C. Architecture Evaluation

After $stage_{final}$, we use the final discretized network as the best network G_{final} and evaluate this network, where the hyper-parameter settings are the same DARTS. On the CIFAR10 and CIFAR100 datasets, the evaluated network has a cell number of 20, the initial number of channels is set to 36, the optimizer is chosen to use SGD, the learning rate and learning rate decay are set to 0.025 and 3e-4, respectively, all the training dataset was used to train and the specialized testing dataset was used to test, and choosing decreasing learning rate strategy of cosine descent, and the number of training iterations was 600 epochs. As in DARTS and other papers [7], [8], [10], [15], data augmentation and some tricks were performed, including slicing, flipping, and cutout [10] of the dataset, while using a drop path with 0.2 probability [24] and auxiliary towers with weight 0.4.

TABLE II. Different Search Settings in Search Strategy (on CIFAR10/CIFAR100)

Stage	Number of Cell	Epoch	Number of Train Data
$stage_{init}$	4	5	10000
$stage_{mid}$	6	5	10000
$stage_{final}$	8	15	20000

The results of the model evaluation are compared with the results in other NAS works as shown in 0The final network g'_{best} obtained in $stage_{final}$ achieves the best test error of 2.6% on CIFAR10 and 16.29 % on CIFAR100 after the model evaluation of 2.5 GPU Days on RTX 2080Ti, slightly better than the original DARTS, which gets 3% test error on CIFAR10 and 19.5% test error on CIFAR100. and the time consumption is also reduced by 50% compared to the original DARTS. Compared with other NAS approaches, ME-DARTS provides a different multi-stage idea from the perspective of evolutionary algorithms, and this idea can be used in other network architecture search algorithms.

D. Validity Verification

1) Validation of Multi-stage search: In order to further verify the validity of EDARTS, we obtain the networks g'_{init} and g'_{mid} from two populations of pop_{init} and pop_{mid}, respectively, according to the inheritance relationship. g'_{mid} inherits the network architecture parameters from g'_{init} and g'_{best} inherits the network architecture parameters from g'_{mid}. We evaluate the performance of these three network structures on CIFAR10 and CIFAR100 respectively, and the experimental settings are referred to the settings in the architecture evaluation in Section IV.C. From the experimental results shown in TABLE IV. , as the algorithm stage progresses, the performance of the network gradually improves.

2) Validation of dataset approximation： In ME-DARTS, we use D_{sample} and D_{low} for searching in all stage. The purpose of using D_{sample} and D_{low} is to accelerate the algorithm and improve its efficiency, and the relative performance and ranking among networks will not change significantly. In order to verify the validity of the partial-sample dataset. We random sampled 10 different networks, conducted

TABLE III. RESULTS COMPARED TO OTHER NAS METHODS

Architecture	Param (M)	Search Cost (GPU days)	Method	Test Err. (%)	
				CIFAR10	CIFAR100
DenseNet-BC [25]	25.6	-	manual	3.46	17.18
NASNet-A [9]	3.3	1800	RL	2.65	-
ENAS [7]	4.6	0.5	RL	2.89	-
AmoebaNet-B [8]	2.8	3150	evolution	2.55±0.05	-
E2EPP [19]	-	8.5	evolution	5.3	22.02
PNAS [26]	3.2	225	SMBO	3.41±0.09	-
DARTS [5]	3.3	1.5	gradient	2.87	17.76
SNAS [23]	2.8	1.5	gradient	2.85	-
ProxylessNAS [16]	5.7	4	gradient	2.08	-
P-DARTS CIFAR10 [4]	3.4	0.3	gradient	2.5	16.55
P-DARTS CIFAR100 [4]	3.6	0.3	gradient	2.62	15.92
ASAP [27]	2.5	0.2	gradient	2.49±0.04	15.6
PC-DARTS [18]	3.6	0.1	gradient	2.57±0.07	-
ME-DARTS CIFAR10	**3.5**	**0.55**	**gradient + evolution**	**2.6**	**16.29**
ME-DARTS CIFAR100	**4.05**	**0.45**	**gradient + evolution**	**2.92**	**15.68**

validation experiments on the CIFAR10 dataset. In the experimental setup, the number of cells is set to 8, the epoch is set to 100, the batch size is set to 96.

a) Validation of D_{sample}: CIFAR10 has 50,000 training images. According to the number of samples in D_{sample}, we have 5 different datasets D_{sample}^{10000}, D_{sample}^{20000}, D_{sample}^{30000}, D_{sample}^{40000} and D_{origin}. D_{origin} represents the original dataset. Each network is repeated for 5 training experiments in different dataset groupings, and the average accuracy of each network is calculated under D_{sample}. The average accuracies of each network under the adjacent number samples are constructed as an ordered pair $p_i(acc_m, acc_n), i \in [1,10]$, the distribution of all ordered pairs in the plane is shown in Fig. 5.

b) Validation of D_{low}: Similarly, we use D_{low} for search in all search stages of ME-DARTS. According to the different resolutions, we have 2 different datasets D_{low} and D_{origin}. In D_{low}, the resolution of images is changed from $32 \times 32 \times 3$ to $16 \times 16 \times 3$. The cutout length [28] in the training process is changed from 16 to 8. Each network repeats the training experiment 5 times in each experimental grouping respectively, and the average accuracies of each network under different resolution groupings is calculated. The accuracies of each network under D_{low} and D_{origin} are constructed as an ordered pair $p_j(acc_l, acc_s), j \in [1,10]$, and the distribution of all ordered pairs is shown in Fig. 6.

TABLE IV. EVALUATION ACCURACY OF EACH STAGE

Architecture	Test Err. (%)	
	CIFAR10	CIFAR100
g'_{best}	2.6	15.68
g'_{mid}	2.82	16.34
g'_{init}	3.32	17.18

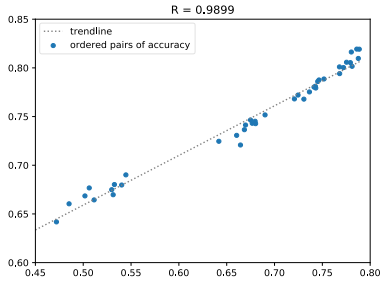

Fig. 5. The horizontal axis is the accuracy obtained by training the network under a partial sample data set, and the vertical axis is the accuracy obtained by training the network under a large sample data set. Each point is an ordered pair obtained by training the same network under the number of adjacent samples, and the dashed line is the trend line of these points with a correlation coefficient of 0.9899.

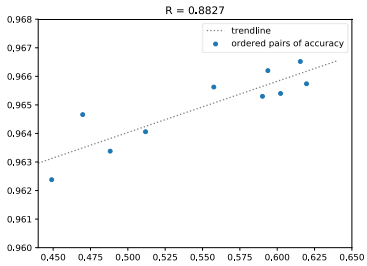

Fig. 6. The horizontal axis is the accuracy obtained by training the network at. low resolution, and the vertical axis is the accuracy obtained by training the network at standard resolution, each point is an ordered pair obtained by training the same network at low and standard resolutions. The dashed line is the trend line of these points, corresponding to a correlation coefficient of 0.8827.

V. CONCLUSIONS

In this paper, we perform neural network architecture search in the direction of differentiable architecture search. To address the drawback that DARTS tends to fall into structural local optima, we propose the ME-DARTS algorithm, in which we introduce an evolutionary strategy and a progressive multi-stage search strategy, while introducing low-resolution dataset approximation computation and partial sample dataset approximation computation and to select quality network individuals more efficiently. From the experimental results, we achieve better results than the original DARTS, and reduce the computational time consumption. Compared with other DARTS-related work, our algorithm extends the differentiable architecture search from an evolutionary perspective and does not require modifying the network architecture and search space. In the future direction of our work, we will try to introduce more evolutionary related approaches, such as:

- genetically encoding the network architecture, designing a better mechanism for individual selection of populations .
- improving the generalizability of multi-stage progressive strategies that can be introduced into other algorithms besides DARTS and be applicable to more tasks, such as target detection, semantic segmentation, etc.

REFERENCES

[1] M. R. Joglekar, C. Li, M. Chen, T. Xu, X. Wang, J. K. Adams, P. Khaitan, J. Liu, and Q. V. Le, "Neural input search for large scale recommendation models," in Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, 2020, pp. 2387–2397.

[2] H. Xu, L. Yao, W. Zhang, X. Liang, and Z. Li, "Auto-fpn: Automatic network architecture adaptation for object detection beyond classification," in Proceedings of the IEEE International Conference on Computer Vision, 2019, pp. 6649–6658.

[3] G. Ghiasi, T.-Y. Lin, and Q. V. Le, "Nas-fpn: Learning scalable feature pyramid architecture for object detection," in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2019, pp. 7036–7045.

[4] X. Chen, L. Xie, J. Wu, and Q. Tian, "Progressive differentiable architecture search: Bridging the depth gap between search and evaluation," in Proceedings of the IEEE International Conference on Computer Vision, 2019, pp. 1294–1303.

[5] H. Liu, K. Simonyan, and Y. Yang, "Darts: Differentiable architecture search," in International Conference on Learning Representations, 2018.

[6] R. Luo, F. Tian, T. Qin, E. Chen, and T.-Y. Liu, "Neural architecture optimization," in Advances in Neural Information Processing Systems, 2018, pp. 7816–7827.

[7] H. Pham, M. Y. Guan, B. Zoph, Q. V. Le, and J. Dean, "Efficient neural architecture search via parameters sharing," in International Conferenceon Machine Learning, 2018, pp. 4092–4101.

[8] E. Real, A. Aggarwal, Y. Huang, and Q. V. Le, "Regularized evolution for image classifier architecture search," in Proceedings of the AAAI Conference on Artificial Intelligence, vol. 33, 2019, pp. 4780–4789.

[9] Zoph and Q. V. Le, "Neural architecture search with reinforcement learning," arXiv preprint arXiv:1611.01578, 2016.

[10] C. Liu, L.-C. Chen, F. Schroff, H. Adam, W. Hua, A. L. Yuille, and L. Fei-Fei, "Auto-deeplab: Hierarchical neural architecture search for semantic image segmentation," in 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2019, pp. 82–92.

[11] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in ICLR 2015 : International Conference on Learning Representations 2015, 2015.

[12] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016, pp. 770–778.

[13] E. Real, S. Moore, A. Selle, S. Saxena, Y. L. Suematsu, J. Tan, Q. Le, andA. Kurakin, "Large-scale evolution of image classifiers," arXiv preprintarXiv:1703.01041, 2017.

[14] Z. Zhong, J. Yan, W. Wu, J. Shao, and C.-L. Liu, "Practical block-wise neural network architecture generation," in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2018, pp. 2423–2432.

[15] Zoph, V. Vasudevan, J. Shlens, and Q. V. Le, "Learning transferable architectures for scalable image recognition," in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2018, pp. 8697–8710.

[16] H. Cai, L. Zhu, and S. Han, "Proxylessnas: Direct neural architecture search on target task and hardware," in International Conference on Learning Representations, 2018.

[17] Deb, S. Agrawal, A. Pratap, and T. Meyarivan, "A fast elitist non-dominated sorting genetic algorithm for multi-objective optimisation: Nsga-ii," in PPSN VI Proceedings of the 6th International Conferenceon Parallel Problem Solving from Nature, vol. 6, no. 6, 2000, pp. 849–858.

[18] Xu, L. Xie, X. Zhang, X. Chen, G.-J. Qi, Q. Tian, and H. Xiong, "Pc-darts: Partial channel connections for memory-efficient differentiable architecture search," arXiv preprint arXiv:1907.05737, 2019.

[19] 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 Transactions on Evolutionary Computation, vol. 24, no. 2, pp. 350–364, 2020.

[20] H. Liang, S. Zhang, J. Sun, X. He, W. Huang, K. Zhuang, and Z. Li, "Darts+: Improved differentiable architecture search with early stopping," arXiv preprint arXiv:1909.06035, 2019.

[21] X. Chu, T. Zhou, B. Zhang, and J. Li, "Fair darts: Eliminating unfair advantages in differentiable architecture search," in Proceedings of the European Conference on Computer Vision (ECCV), 2020, pp. 465–480.

[22] A. Brock, T. Lim, J. M. Ritchie, and N. Weston, "Smash: one-shot model architecture search through hypernetworks," arXiv preprint arXiv:1708.05344, 2017.

[23] S. Xie, H. Zheng, C. Liu, and L. Lin, "Snas: stochastic neural architecture search," in International Conference on Learning Representations, 2018.

[24] G. Larsson, M. Maire, and G. Shakhnarovich, "Fractalnet: Ultra-deep neural networks without residuals." arXiv preprint arXiv:1605.07648, 2016.

[25] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2017, pp. 4700–4708.

[26] C. Liu, B. Zoph, M. Neumann, J. Shlens, W. Hua, L.-J. Li, L. Fei-Fei, A. Yuille, J. Huang, and K. Murphy, "Progressive neural architecture search," in Proceedings of the European Conference on Computer Vision (ECCV), 2018, pp. 19–34.

[27] A. Noy, N. Nayman, T. Ridnik, N. Zamir, S. Doveh, I. Friedman,R. Giryes, and L. Zelnik-Manor, "Asap: Architecture search, anneal and prune," arXiv preprint arXiv:1904.04123, 2019.

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