A Classification Surrogate Model based Evolutionary Algorithm for Neural Network Structure Learning

Wenyue Hu, Aimin Zhou, and Guixu Zhang
Shanghai Key Laboratory of Multidimensional information Processing
School of Computer Science and Technology, East China Normal University, Shanghai, China
51184506013@stu.ecnu.edu.cn, {amzhou,gxzhang}@cs.ecnu.edu.cn

***Abstract*—Designing neural networks often requires a large number of artificial intelligence experts. However, such manual processes are time-consuming and require numerous resources. In this paper, we try to search neural network structures automatically for the image classification task. Moreover, considering the huge computational cost of neural architecture search (NAS), we attempt to apply a classification surrogate model based multi-objective evolutionary algorithm to search neural network architectures (CSMEA-Net). The algorithm combines two objectives, i.e., minimizing the validation error and the computational complexity measured by the number of floating-point operations (FLOPs) to achieve Pareto Optimality. In addition, we improve the components of the cell-based search space. The performance of network architectures discovered by our method is evaluated on CIFAR-10 and CIFAR-100 datasets. The experimental results show that the proposed approach can find a higher-performance neural network architecture compared with both hand-crafted as well as automatically-designed networks.**

***Index Terms*—convolutional neural network, neural architecture search, multi-objective evolutionary algorithms, image classification**

I. Introduction

Current convolutional neural networks (CNNs) have a great number of hyper-parameters and parameters in convolutional and pooling layers, which are inefficient to be set manually by human experts. Therefore, some methods that optimize the hyper-parameters of CNNs, such as Grid Search (GS), Random Search (RS) [1], Bayesian-based Gaussion Process (BGP) [2], and Sequential Model-Based Global Optimization (SMBO) [3] have come up gradually. However, these architecture-search methods are only used for optimizing some pre-defined parameters of a fixed CNN structure, such as the kernel size and the number of channels in each layer. It is not flexible because it still needs engineers to design the whole network structure manually. Later, neural architecture search (NAS) [4], which can automatically search for the optimal architecture has been drawing the growing attention of researchers. The widely used methods contain reinforcement learning (RL) [5], evolutionary algorithm (EA) [6], [7], and gradient descent (GD) [8]. Though many architectures found in these automatic search algorithms have already achieved greater performance than state-of-the-art manually-designed CNNs, there are two limitations of current approaches. Firstly, some RL and EA methods require massive computational resources. Second, the gradient-based method just focuses on a single objective, which minimize the error metric in a specific task.

In this paper, we present a novel NAS method to address the aforementioned problems. To balance the limited computational resources and the predictive performance in a specific task, we use a multi-objective evolutionary algorithm (MOEA) as the searching approach to evolve and optimize the network architecture. We also attempt to integrate K-Nearest Neighbors (KNN) [9] into the selection procedure of a MOEA in order to reduce the computational cost and speed up the training time. The main contributions of our work are demonstrated as follows:

- We develop a discrete scheme to encode a neural network structure and reproduce the next population, which takes inspiration from the classification based preselection (CPS) strategy for multi-objective evolutionary optimization [10]. We use KNN as a classification surrogate model to learn and evaluate the discrete chromosomes' performances.
- We improve the current neural architecture search space by adding several novel components to the NASNet [5] search space in order to further enhance the accuracy of the ultimate model. Besides, we redesign the architecture of the final neural network. We also use a new nonlinearity activation function h-swish [11] to substitute ReLU for increasing training accuracy and computational speed.
- Within only 4 days on 1 GPU, our method discovers a competitive neural network architecture with respect to higher accuracy and less resource consumption compared with hand-designed networks such as DenseNet [12], single-objective and multi-objective EA-based networks, and RL-based networks.

The rest of this paper is organized as follows. Section II describes the background and related work about image classification and the development of neural networks. Section III presents our method of network representation and optimization, which includes the proposed search space and

This work is supported by the National Nature Science Foundation of China (Nos. 61731009, 61673180) and the Science and Technology Commission of Shanghai Municipality (No. 19511120600). (Corresponding author: Guixu Zhang.)

978-1-7281-6926-2/20/$31.00 ©2020 IEEE

how to encode and decode a neural network. Section IV demonstrates a classification surrogate model based multi-objective evolutionary algorithm to search a set of CNN architectures. Section V presents the experimental setting and the results of the proposed algorithm. Then we analyze these results. In section VI, we sum up this paper and discuss future work.

II. RELATED WORK

Image classification can be described as the task of classifying images into several pre-defined classes. It is an elemental application in computer vision, which lays the foundation for other computer vision tasks such as image restoration [13] and object detection [14].

There are usually two-stage approaches to deal with the image classification problem. First, extracting features from images as the input of a trainable classifier. The design of the feature extraction stage has a great influence on the performance of classification tasks, which traditionally is regarded as a challenging task. CNNs were useful architectures to overcome these challenges. Moreover, Deep CNNs (DCNNs) that were supported by large data sets and GPUs brought the renaissance to neural networks. Krizhevsky et al. [15] used a DCNN to achieve first place in the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) 2012.

Szegedy et al. [16] proposed an inception DCNN architecture, GoogLeNet, which won both the ImageNet classification and detection challenges in 2014. He et al. [17] put forward deep residual networks (ResNets), which showed state-of-the-art accuracy for some challenging recognition tasks on ImageNet and MS COCO competitions. Though ResNets had more than 100-layer depth, ResNets solved the degradation problem by identity mappings. Residual Inception networks [18] combining residual connections and inception architecture achieved better performance than other similarly Inception Networks on the ImageNet classification (CLS) challenge.

Generating neural network architectures automatically ha aroused great interest in recent years. Generally speaking these methods can be divided into reinforcement learnin (RL), evolutionary algorithm (EA), gradient descent (GD) an some other approaches.

The MetaQNN method [19] applied a Q-learning method to search connections in convolutional layers, pooling layers, and fully connected layers. Zhong et al. [20] also used a Q-learning strategy to explore the design of a computational block, which was then repeated several times to build an entire network. Zoph and Le [21] applied a policy gradient method to train a recurrent neural network controller to generate the whole network at the same time. NASNet [5] repeated several normal cells and reduction cells to construct a neural network. Considering the trade-off between prediction accuracy and energy consumption, Hsu et al. [22] extended the NASNet method to the multi-objective domain. However, multiple linear optimizations of different scalarized objectives were confirmed to be not as efficient as simultaneous approaches [23].

Multi-objective evolutionary algorithms have been little used for NAS. Kim et al. [24] proposed Neuro-Evolution with Multi-objective Optimization (NEMO), an algorithm utilizing NSGA-II [23] to minimize inference time and error metric; however, their methods only explored some hyper-parameters and the number of output channels of each layer on a small set of some architectures. The encoding way of NSGA-Net [25] was similar to the binary encoding scheme of Genetic CNN [7], and NSGA-Net used NSGA-II as the searching algorithm. However, the computational space of binary encoding was square about the number of nodes. Dong et al. [26] proposed Platform-aware Progressive search for Pareto-optimal Net (PPP-Net), which automatically generated neural networks with a pre-dened number of replicated blocks. Elsken et al. [12] demonstrated an evolutionary algorithm LEMONADE for multi-objective architecture search and used network morphisms operators to generate children networks.

Most of the aforementioned methods require huge computational resources because they need to train and validate thousands of neural network. A possible way of reducing such computational burden is to estimate an architecture's performance by learning curve extrapolation (e.g., [3]). However, there are barely surrogate models in MOEAs applied to NAS. We attemp to speed up evaluation by using a surrogate model to predict the performance of a trained model.

III. REPRESENTATION AND OPTIMIZATION OF NEURAL NETWORK STRUCTURE LEARNING

This section introduces the overall procedure of neural network structure optimization, including the definition of operation search space, and the encoding and decoding strategy to represent a neural network. Our method consists of two stages, i.e., architecture search stage and architecture evaluation stage. The flow diagram of the neural network structure learning is shown in Fig. 1.

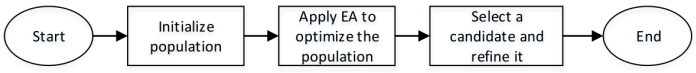

Fig. 1: The flow diagram of network structure optimization

In the first stage, all individuals in the first generation are randomly initialized. We propose an approach to encode each neural network architecture into a discrete chromosome, which represents an individual in a population. Then we define some useful operations in the search space so as to make the evolutionary algorithm explore network architectures more efficiently. Each individual can be decoded to a normal convolutional cell and a reduction convolutional cell, which are the compositions of a whole neural network. Then we train every neural network roughly, that is, training fewer epochs in the first stage. The test accuracy is regarded as the fitness value to indicate the performance of each individual. Returning these evaluation results to the multi-objective evolutionary

algorithm, and the algorithm applies crossover and mutation to generate the next population. These steps in the process will be executed circularly until exceeding a given maximum number of the function evaluations and obtain a set of network candidates with great performance. In the second stage, we pick an individual in the final population, train it from scratch and report its performance on the test set.

A. Operation search space

Based on the cell-based search space, we explore two types of computation cells as the building blocks of the overall architecture. The normal cells preserve the image size of the input feature map, while the reduction cells reduce the image size by a factor of two. We define the computational operations in TABLE I.

TABLE I: Computational operations of search space

index	operation
0	zero
1	3×3 max pooling
2	3×3 average pooling
3	skip connection
4	3×3 depthwise-separable convolution
5	5×5 depthwise-separable convolution
6	3×3 dilated convolution
7	5×5 dilated convolution
8	1×7 then 7×1 convolution
9	3×3 linear bottleneck and inverted residual convolution

First of all, in order to reduce computation cost and model size, we replace the traditional convolution layers with the depthwise separable convolution, the linear bottleneck and inverted residual structure, and the dilated convolution.

The depthwise separable convolution can be regarded as two separate layers: a depthwise convolution for spatial filtering and a 1×1 pointwise convolution for feature generation. Meanwhile, we use linear bottleneck and inverted residual convolution which applies linear convolution to expand a low-dimensional representation to a high dimension, to filter it with a depthwise convolution, and to project the features back to a low-dimensional representation. Besides, we utilize dilated convolutions to expand the receptive field without losing resolution or coverage. Because the h-swish activation function has the characteristics of no upper bound and lower bound, smoothness and no monotony, which is better than ReLU in deep neural networks, we substitute the ReLU function with h-swish in deeper layers to improve the accuracy of neural networks. The h-swish and ReLU6 are defined as:

$$\text{h-swish}[t] = t\frac{\text{ReLU6}(t+3)}{6} \tag{1}$$

$$\text{ReLU6} = min(\max(0, t), 6) \tag{2}$$

Each cell is organized by several nodes to form a directed acyclic graph (DAG). Assume that every node is a feature map in neural networks and every connected edge is related to several pre-defined operations. So each edge will perform an operation on its connected node.

Especially, there are three characteristics of the final network architecture. Firstly, we connect the two types' cells by such a pattern: each cell gets the outputs of the two previous cells as input. Reduction cells locate at the $\lfloor \frac{1}{3} \rfloor$ and $\lfloor \frac{2}{3} \rfloor$ depth of the final neural network, and the rest are normal cells. Secondly, we embed the Squeeze-and-Excitation module into every cell to use global information in order to selectively emphasize the important feature map and surppress less useful information for the purpose of improving the representational power of a network. Last, motivated by deep pyramidal residual networks, we gradually increase the channel dimension of each cell instead of down-sampling sharply.

B. Encoding and decoding strategy

Assume that a neural network architecture u is composed by m cells. Each cell has n nodes and we can encode a cell to a chromosome of length $4n$, such as x is a string of $[x_1, x_2, x_3, x_4, ..., x_{4n-3}, x_{4n-2}, x_{4n-1}, x_{4n}]$. For example, using $[x_{4i-3}, x_{4i-2}]$, $[x_{4i-1}, x_{4i}]$ denote two inputs of the i-th node. The x_{4i-2} and x_{4i-1} value range from 0 to 9 denoting 10 pre-defined operations, while x_{4i-2} and x_{4i-2} range from 0 to i denoting the previous nodes before node i. x_{4i-3} represents the index of the first operation op_{4i-3}, and x_{4i-2} represents the index of the first input feature map. As shown in Eq. (3), we apply add operation to the two inputs $op_{4i-3}(x_{4i-2})$ and $op_{4i-1}(x_{4i})$.

$$node[i] = add(op_{4i-3}(x_{4i-2}), op_{4i-1}(x_{4i})) \tag{3}$$

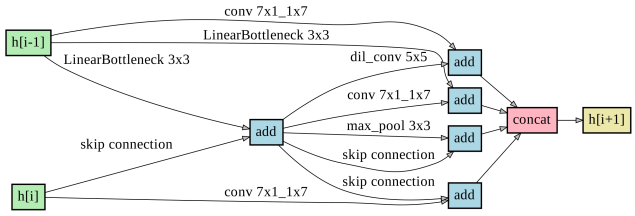

Fig. 2: Normal cell learned on CIFAR-10

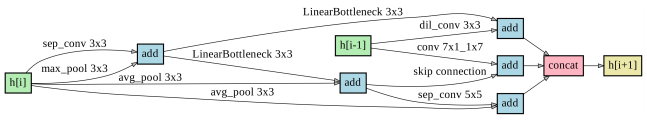

Fig. 3: Reduction cell learned on CIFAR-10

Fig. 2 and Fig. 3 show an example of the normal convolutional cell and reduction convolutional cell. $h[i-1]$ and $h[i]$ (green color nodes) are the output of previous cells. The inputs of feature maps (blue color nodes) are the outputs of previous nodes. Each edge (line with arrow) having an operation name annotated above the line indicates an operation of the pre-defined search space as TABLE I described. In decoding scheme, mapping every pairwise tuples as a group of input.

For example, for an encoding string, the pairwise tuple [[3, 1], [9, 0]] can be decoded to [[skip connection, 1], [LinearBottleneck 3×3, 0]], which means applying the operation 3×3 linear bottleneck and inverted residual convolution to $h[i]$ and applying skip connection to $h[i-1]$. Then more feature maps are constructed by taking them as two new inputs and adding them together. Finally, the feature maps that remain unused as input are concatenated to form the output of the cell.

As shown in Eq. (4), one of the objectives in our serach algorithm is to minimize the fitness value and the other one is to minimize the complexity of the neural network. Training each neural network u to obtain the tness function value $fit_1(x) = NN(u)$ and the complexity $fit_2(x) = Flops(u)$.

$$\begin{aligned} &\text{minimize } fit_1(x) = NN(u) \\ &\text{minimize } fit_2(x) = Flops(u) \end{aligned} \tag{4}$$

IV. Evolutionary Algorithm for Neural Network Search

In this section, we present the multi-objective evolutionary algorithm based on preselection [10] to evolve a set of competitive network structures. CPS-MOEA [10] particularly focuses on real-parameter optimization. However, in view of the representation of neural network architecture, we modify the reproduction operators in CPS-MOEA in order to make them more suitable for discrete chromosomes.

In traditional evolutionary algorithms, the selection operator depends on the real calculated objective values of offspring in each generation to select a certain number of promising candidates. Similarly, the validation accuracy of every candidate network can be regarded as an indicator to reflect the offspring performance in NAS. Usually, generating more offspring solutions is more likely to find the best solution. However, training a deep neural network requires a large amount of time and computational resources. So the fitness evaluation of an individual is extremely expensive. We attempt to use a classifier as the surrogate model to learn the approximate performance of each network architecture for eliminating bad offspring and retaining good ones.

A. Classifier Training

In each generation, our algorithm divides the population C into two clusters and maintains two additional populations: C_+ consists of $\lfloor \frac{N}{2} \rfloor$ positive solutions, and C_- contains $\lfloor \frac{N}{2} \rfloor$ negative solutions to train a classifier. In multi-objective optimization, Pareto domination can be used to judge the good and bad solutions: labelling a non-dominated solution x as $Class(x) = +1$; labelling a dominated solution x as $Class(x) = -1$.

In this paper, we choose the K-Nearest Neighbor (KNN) as the classifier to solve the classification problem. Let $\{\langle x,\ \text{Class}\ (x)\rangle\}$ be a set of training data, where x is a feature vector representing for the encoding of network structure and $Class(x)$ is its corresponding $label \in \{-1, +1\}$. Their relationship can be denoted as $label = Class(x)$. As shown in Eq. (5), The target of KNN is that taking C_+ and C_- as the training dataset to find an approximate relationship $label = \widehat{\text{Class}}(x)$ in order to replace the real relationship. x^i denotes the ith closest feature vector, according to the Euclidean distance.

$$KNN(x) = \begin{cases} +1 & \sum_{i=1}^{K} Class\left(x^i\right) \geq 0 \\ -1 & otherwise \end{cases} \tag{5}$$

B. Offspring Reproduction

In Line 8 of Algorithm 1, the next generation is created by the slight mutation and two-point crossover to transform architectures of existing candidates. The two-point crossover operator illustrated in Fig. 4 is a specific case of a n-point crossover technique. Two random points are chosen on the parental chromosomes and the genetic material is exchanged at these points to exchange information between individuals during the search. The mutation operator removes one bit of a chromosome and replaces it by a different bit in a randomly chosen position. In our search space, the mutation can change the current operation or feature map in each cycle by choosing one of the two at random. For a better explanation, we name them as feature map mutation and operation mutation, an example shown in Fig. 5. The feature map mutation will replace the chosen feature map with another feature map in the same cell as the input of the next node. The operation mutation will modify the existing operation with a random choice from the pre-defined list of operations.

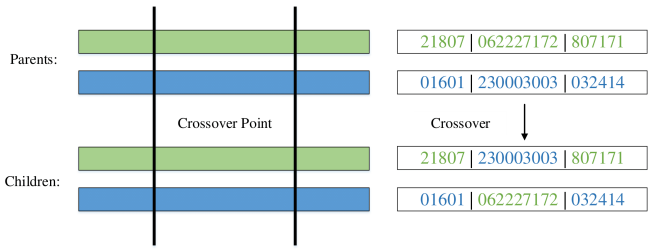

Fig. 4: Illustration of the two-point crossover

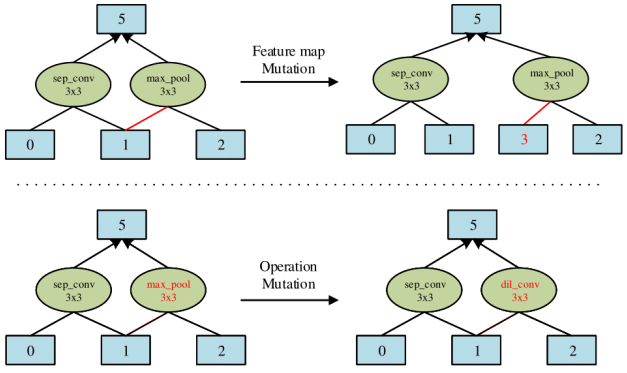

Fig. 5: Illustration of the two mutation operators

C. Algorithm Framework

The whole algorithm framework is shown in Algorithm 1. In Line 3, the algorithm will stop when the current number of generation g exceeds a given maximum number of generation g_{max}. In Line 5, $C_+ = NDS\left(C, \lfloor \frac{N}{2} \rfloor\right)$ means using the nondominated sorting scheme [23] to choose the best $\lfloor \frac{N}{2} \rfloor$ solutions from the population C of size N and reserve them in C_+, and C_- contains the remaining half. In Line 8-16, a size of M offspring solutions Y are generated by the reproduction operator. Afterward, KNN is used to predict their corresponding labels. Then the individuals with $label = +1$ are selected as the offspring solutions and stored in set G. In Line 18-21, the newly generated non-dominated G_+ and dominated G_- solutions are used to update C_+ and C_- respectively. In Line 24 and 25, picking an individual from the Pareto Frontier of the final population, and then refining it to get the fitness function value.

Algorithm 1 Main framework of CSMEA-Net

1: Initialize the population $C = \{x^1, x^2, \ldots, x^N\}$ with the proposed gene encoding strategy;
2: Set the current generation $g = 1$;
3: **while** $g \leq g_{max}$ **do**
4: Decode each x^i in C and evaluate it to get the fitness;
5: Set $C_+ = NDS\left(C, \lfloor \frac{N}{2} \rfloor\right)$ and $C_- = C \backslash C_+$;
6: Train a classifier $k = \widehat{\text{Class}}(x)$ with dataset $C_+ \cup C_-$;
7: Set $G = \emptyset$;
8: **for** x in C **do**
9: Generate M offspring $Y = \{y^1, y^2, \ldots, y^M\}$;
10: Set $Y' = \left\{ y \in Y | \widehat{\text{Class}}(y) = 1 \right\}$;
11: **if** Y' is empty **then**
12: $Y' = Y$;
13: **end if**
14: Randomly choose $y \in Y'$ as the offspring solution;
15: Set $G = G \cup \{y\}$;
16: **end for**
17: Update $C = NDS\left(C \cup G, N\right)$;
18: Set G_+ be the non-dominated solutions in G;
19: Set G_- be the dominated solutions in G;
20: Update $C_+ = NDS\left(C_+ \cup G_+, \lfloor \frac{N}{2} \rfloor\right)$;
21: Update $C_- = NDS\left(C_- \cup G_-, \lceil \frac{N}{2} \rceil\right)$;
22: Update $g = g + 1$;
23: **end while**
24: Pick an individual x^i from the Pareto Frontier of the final population C;
25: Refine x^i to get the $fit(x^i)$
26: **return** $fit(x^i)$;

V. Experiments and Results

A. Benchmark Dataset

We choose CIFAR-10 and CIFAR-100 as the benchmark datasets to test our method. The CIFAR-10 dataset consists of 60,000 color images with the size of 32×32 in 10 classes. 80 percent of the whole dataset are used for training, and the rest are used for testing. The CIFAR-100 dataset has the same image size and format as the CIFAR-10 dataset, while it contains 100 classes.

B. Implementation Details

The experiments are carried out in two stages: architecture search stage which trains every architecture to get a rough test result; architecture evaluation stage which trains the selected network with more epochs. We set the number of nodes to 5 and the number of operations to 10 in each cell. The initial population is generated by uniform random sampling. Considering the convention of evolutionary algorithm and the high computational cost of our experiment, the population size is set as $N = 40$ and the maximum number of generations g_{max} is 30. Based on empirical experience, the number of nearest points K in KNN is set as 3. The number of generated offspring solutions for each individual is set as $M = 3$ in the preselection procedure of CPS-MOEA.

When training neural networks during searching stage, we use standard stochastic gradient descent (SGD) back-propagation algorithm and a cosine annealing learning rate schedule [27]. The initial learning rate is 0.025, the training epochs are 25, and the batch size is 76. In the architecture evaluation stage, we use the same configuration, except increasing the training epochs to 600. We train the selected architecture which is composed of 10 cells from scratch with all the training data and evaluate it on the test set.

C. Results Analysis

Our algorithm aims at minimizing two objectives: the validation accuracy on CIFAR-10 and CIFAR-100; the number of multiply-add operations (FLOPs) that means the total count of operations in the forward pass of each evolved network. We pick the last 4 stages of the bi-objective Pareto Frontiers obtained by CSMEA-Net to plot in Fig. 6, which clearly shows a gradual improvement of the whole population. To compare the network searched from our algorithm with other hand-crafted and automated architectures, we choose the network with the lower classification error from the final Pareto Frontier (the knee point marked with black frame in Fig. 6) and train with the entire CIFAR-10 and CIFAR-100 training set. The chosen neural network results in 2.51% test error on the CIFAR-10 test set with 2.42 Millions of parameters and 16.28 MFLOPs. Fig. 7 plots the classification accuracy of our model within 600 epoches. It can be seen that the accuracy quickly increases before 100 epochs. Between 100 and 600 epochs, the accuracy gradually enhances and has a little range fluctuation. Finally, it converges to 97.49 percent.

TABLE II presents a summary that compares our network with state-of-the-art hand-designed and automated methods. During the neural network searching stage, the experiment takes approximately 4 days on a single 2080Ti GPU. Notably, the CSMEA-Net far outweighs the human-designed models such as the highest-performing architectures Wide ResNet and DenseNet in every aspect such as parameters, FLOPs,

TABLE II: Comparison with the best classification error of state-of-the-art networks on CIFAR-10 (C-10) and CIFAR-100 (C-100)

Architecctures	Params	Mult-Adds	Test Error(%)		Search Cost	Search Method
	(M)	(M)	on C-10	on C-100	(GPU days)	
Wide ResNet [28]	36.5	5953	4.17	20.5	-	manual
DenseNet-BC ($k = 40$) [12]	25.6	-	3.47	17.18	-	manual
MetaQNN(top model) [19]	11.2	-	6.92	27.14	100	RL
NAS + more filters [21]	37.4	-	3.65	-	3150	RL
ENAS + cutout [29]	3.3	533	2.75	-	0.5	RL
BlockQNN-S more filter [20]	39.8	-	3.54	18.06	96	RL
NASNet-A + cutout [5]	3.3	532	3.41	-	2000	RL
Darts + cutout [8]	3.3	-	2.76	-	4	gradient-based
AmoebaNet-A + cutout [30]	3.2	533	3.34	-	3150	evolution
GeNet [7]	156	-	7.10	29.05	-	evolution
CGP-CNN(ConvSet) [31]	1.75	-	6.34	-	-	evolution
CGP-CNN (ResNet) [31]	2.64	-	6.05	-	28	evolution
PPP-Net-Baseline [26]	11.39	1364	4.36	17.18	-	multi-evolution
MONAS [32]	-	-	4.34	-	-	multi-evolution
LEMONADE [6]	4.7	-	3.05	-	56	multi-evolution
NSGA-Net (filters=128) [25]	3.3	1290	3.85	20.74	8	multi-evolution
CSMEA-Net	**2.42**	**16.28**	**2.51**	**15.91**	**4**	**multi-evolution**

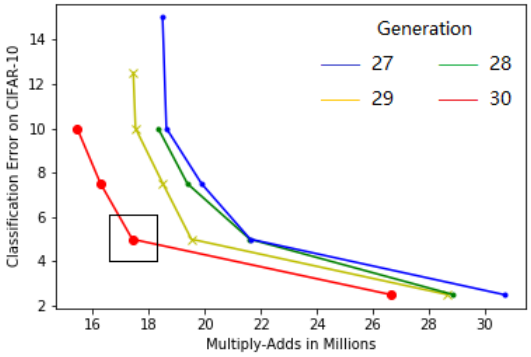

Fig. 6: Progress of the Pareto front of CSMEA-Net during architecture search (The knee point marked with black frame is the chosen neural network to refine.)

and test accuracy. RL-search and EA-generated methods need orders of magnitude much computation resources than our method, which illustrates the necessity of a surrogate model based multi-objecctive evolutionary algorithm. Compared to the gradient-based NAS method such as DARTS, we also have the advantage in test error and parameters. One of the greatest strengths of the multi-objecctive evolutionary algorithm (MOEA) is that we can flexibly choose an architecture from the Pareto Frontier according to different scenarios. For example, if engineers want to construct an efficient mobile architecture, they could pick the neural network with fewer FLOPs. Moreover, the classification surrogate model employed in MOEA can approximate the original objective function and estimate the performances of the candidate solutions. Then based on the estimated fitness values, promising solutions can be identified. Therefore, the surrogate model can help our algorithm to use less computational resources and make our algorithm more fast than other multi-objecctive NAS methods.

Besides, we test our model on CIFAR-100 dataset, which is evolved on CIFAR-10 with the same training setup for evaluating the transferability. The training time takes about 1.5 GPU days. Results shown in TABLE II demonstrate that the architecture found by our method can be transferable to CIFAR-100 and achieves better performance than other architectures.

The decoding architecture is shown in Fig. 2 and Fig. 3, and the encoding chromosome of cell structure is [3, 1, 9, 0, 8, 0, 7, 2, 3, 2, 8, 1, 8, 2, 9, 0, 3, 2, 1, 2, 4, 1, 1, 1, 9, 2, 6, 0, 9, 2, 2, 1, 5, 4, 2, 1, 8, 0, 3, 4], which can be divided into two parts of equal length. The first part [3, 1, 9, 0, 8, 0, 7, 2, 3, 2, 8, 1, 8, 2, 9, 0, 3, 2, 1, 2] can be decoded to a normal cell, and the second part [4, 1, 1, 1, 9, 2, 6, 0, 9, 2, 2, 1, 5, 4, 2, 1, 8, 0, 3, 4] can be decoded to a reduction cell.

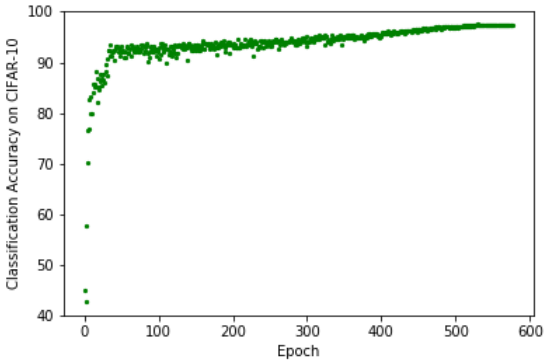

Fig. 7: The classification accuracy of the picked network in each training epoch

VI. Conclusions and Future Work

In this paper, we attempt to use the multi-objective evolutionary algorithm (MOEA) based on a classification surrogate

model for neural architecture search. Our method affords a number of practical benefits. Firstly, the design of neural network architectures can effectively optimize and balance the trade-off of two competing objectives: classification error and computational complexity. Secondly, population based methods can produce a set of individuals in each generation, so experts could select a suitable neural network according to their needs, which is more effective than just optimizing the weighted linear combination of objectives. Last, the classification surrogate model which is implemented into Pareto domination based MOEA framework can filter possible 'negative' offspring solutions to improve the efficiency of MOEA.

We investigate our algorithm's potential on CIFAR-10 and CIFAR-100. The experimental results show that the evolved neural network outperforms state-of-the-art automated and hand-crafted models both in terms of accuracy and model complexity. Thus our method can effectively balance the trade-off between validation accuracy and number of multiply-add operations to discover usable and efficient solutions.

Because evaluating on a large-scale dataset will require massive computational resources, we only test the proposed multi-objective evolutionary algorithm on some commonly used middle-scale benchmarks, not on a large-scale dataset. For future work, we will put effort into efcient tness evaluation methods in order to reduce the search time and the computational resources. Moreover, another future work is that applying the proposed method to other image problems, such as image restoration or object detection.

REFERENCES

[1] J. Bergstra and Y. Bengio, "Random search for hyper-parameter optimization," *Journal of Machine Learning Research*, vol. 13, no. 2, pp. 281–305, 2012.

[2] C. E. Rasmussen and C. K. I. Williams, *Gaussian processes for machine learning*. Adaptive computation and machine learning, MIT Press, 2006.

[3] F. Hutter, H. H. Hoos, and K. Leyton-Brown, "Sequential model-based optimization for general algorithm configuration," in *International conference on learning and intelligent optimization*, pp. 507–523, 2011.

[4] T. Elsken, J. H. Metzen, and F. Hutter, "Neural architecture search: A survey," *Journal of Machine Learning Research*, vol. 20, no. 55, pp. 1–21, 2019.

[5] B. 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*, pp. 8697–8710, 2018.

[6] T. Elsken, J. H. Metzen, and F. Hutter, "Efficient multi-objective neural architecture search via lamarckian evolution," in *International Conference on Learning Representations*, 2019.

[7] L. Xie and A. Yuille, "Genetic CNN," in *Proceedings of the IEEE International Conference on Computer Vision*, pp. 1379–1388, 2017.

[8] H. Liu, K. Simonyan, and Y. Yang, "DARTS: differentiable architecture search," in *International Conference on Learning Representations*, 2019.

[9] D. Coomans and D. L. Massart, "Alternative k-nearest neighbour rules in supervised pattern recognition: Part 1. k-nearest neighbour classification by using alternative voting rules," *Analytica Chimica Acta*, vol. 136, pp. 15–27, 1982.

[10] J. Zhang, A. Zhou, and G. Zhang, "A classification and Pareto domination based multiobjective evolutionary algorithm," in *IEEE Congress on Evolutionary Computation*, pp. 2883–2890, 2015.

[11] A. Howard, M. Sandler, G. Chu, *et al.*, "Searching for MobileNetV3," *IEEE International Conference on Computer Vision*, pp. 1314–1324, 2019.

[12] G. Huang, Z. Liu, V. D. Maaten, *et al.*, "Densely connected convolutional networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2261–2269, 2017.

[13] G. J. van Wyk and A. S. Bosman, "Evolutionary neural architecture search for image restoration," in *International Joint Conference on Neural Networks*, pp. 1–8, 2019.

[14] J. Zhao, X. N. Zhang, H. Gao, J. Yin, M. Zhou, and C. Tan, "Object detection based on hierarchical multi-view proposal network for autonomous driving," in *International Joint Conference on Neural Networks*, pp. 1–6, 2018.

[15] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," in *Conference and Workshop on Neural Information Processing Systems*, pp. 1106–1114, 2012.

[16] C. Szegedy, W. Liu, Y. Jia, *et al.*, "Going deeper with convolutions," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1–9, 2015.

[17] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on ImageNet classification," in *Proceedings of the IEEE international conference on computer vision*, pp. 1026–1034, 2015.

[18] C. Szegedy, S. Ioffe, V. Vanhoucke, and A. A. Alemi, "Inception-v4, Inception-ResNet and the impact of residual connections on learning," in *The Association for the Advancement of Artificial Intelligence*, pp. 4278–4284, 2017.

[19] B. Baker, O. Gupta, N. Naik, and R. Raskar, "Designing neural network architectures using reinforcement learning," in *International Conference on Learning Representations*, 2017.

[20] 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*, pp. 2423–2432, 2018.

[21] B. Zoph and Q. V. Le, "Neural architecture search with reinforcement learning," in *International Conference on Learning Representations*, 2017.

[22] E. Real, S. Moore, A. Selle, S. Saxena, Y. L. Suematsu, J. Tan, Q. V. Le, and A. Kurakin, "Large-scale evolution of image classifiers," in *International Conference on Machine Learning*, vol. 70 of *Proceedings of Machine Learning Research*, pp. 2902–2911, 2017.

[23] K. Deb, S. Agrawal, A. Pratap, and T. Meyarivan, "A fast elitist non-dominated sorting genetic algorithm for multi-objective optimization: NSGA-II," in *Parallel Problem Solving from Nature*, vol. 1917 of *Lecture Notes in Computer Science*, pp. 849–858, 2000.

[24] 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 *Journal of Machine Learning Research: Workshop and Conference Proceedings*, vol. 1, pp. 1–8, 2017.

[25] Z. Lu, I. Whalen, V. Boddeti, *et al.*, "NSGA-Net: neural architecture search using multi-objective genetic algorithm," in *Proceedings of the Genetic and Evolutionary Computation Conference*, pp. 419–427, 2019.

[26] J. Dong, A. Cheng, D. Juan, W. Wei, and M. Sun, "PPP-Net: Platform-aware progressive search for pareto-optimal neural architectures," in *International Conference on Learning Representations*, 2018.

[27] I. Loshchilov and F. Hutter, "SGDR: Stochastic gradient descent with warm restarts," in *International Conference on Learning Representations*, 2017.

[28] S. Zagoruyko and N. Komodakis, "Wide residual networks," in *British Machine Vision Conference*, 2016.

[29] H. Pham, M. Y. Guan, B. Zoph, Q. V. Le, and J. Dean, "Efficient neural architecture search via parameter sharing," in *International Conference on Machine Learning*, vol. 80 of *Proceedings of Machine Learning Research*, pp. 4092–4101, 2018.

[30] 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, pp. 4780–4789, 2019.

[31] M. Suganuma, S. Shirakawa, and T. Nagao, "A genetic programming approach to designing convolutional neural network architectures," in *Proceedings of the Genetic and Evolutionary Computation Conference*, pp. 497–504, 2017.

[32] C.-H. Hsu, S.-H. Chang, J.-H. Liang, and othera, "MONAS: Multi-objective neural architecture search using reinforcement learning," *Computing Research Repository*, vol. abs/1806.10332, 2018.