A Memetic Algorithm for Evolving Deep Convolutional Neural Network in Image Classification

1st Junwei Dong
College of Computer Science
Chongqing University
Chongqing, China
JWDong@cqu.edu.cn

2nd Liangjie Zhang
College of Computer Science
Chongqing University
Chongqing, China
LJZhang@cqu.edu.cn

3rd Boyu Hou
College of Computer Science
Chongqing University
Chongqing, China
byhou@cqu.edu.cn

4th Liang Feng
College of Computer Science
Chongqing University
Chongqing, China
liangf@cqu.edu.cn

***Abstract*—As evolutionary algorithms (EAs) are robust to the problem formulation and easy to use, there is a growing interest in designing EAs for automated neural architecture search in recent years. In particular, EvoCNN is a recently proposed evolutionary algorithm to automate the configuration of a deep Convolutional Neural Network (CNN) for image classification. Its efficacy has been confirmed against 22 existing algorithms for CNN configuration, on the widely used image classification tasks. However, despite the success enjoyed by this method, we note that there are several limitations existed in this method. For example, only chain structured network is considered for evolution. Further, there are many decision variables, which is computational expensive. In this paper, we embark a study on evolutionary neural architecture search by proposing a memetic algorithm (MA), with the aim of addressing the problems mentioned above. Particularly, first of all, besides evolving the chain structured network, local search is designed for multi-branch network search. Next, to reduce the network parameters for optimization, we focus on the architecture search only on the convolutional layers. Moreover, based on a recent hypothesis in the literature, the network evaluation is conducted based on only the early training process in our proposed MA. To confirm the efficacy of the proposed method, comprehensive empirical studies are conducted against EvoCNN for NAS, on the commonly used image classification benchmarks.**

***Index Terms*—Memetic algorithm, neural architecture search, image classification, convolutional nefural network.**

I. Introduction

In recent years, deep convolutional neural networks (CNNs) has achieved considerable success in many image analysis tasks, such as image classification [1]–[5], object detection [6]–[8], and semantic segmentation [9]–[11], etc. CNNs can learn low/mid/high level features from a specific benchmark through alternating convolutional layers and pooling layers. Compared to traditional feedforward neural networks with similarly-sized layers, CNNs have much fewer connections and parameters [1]. In the literature, with the development of CNNs, many successful architectures have been proposed, such as VGGNet [2], GoogleNet [3], ResNet [4], and DenseNet [5] etc. Moreover, these smartly designed networks further reduced the large quantities of connections and parameters, and achieved superior learning performance in many real world applications [1], [6], [11]. However, despite the great successes enjoyed by these various CNN architectures, mostly of them are developed manually by human experts, which is a time-consuming and error prone process. As different applications or tasks may require unique CNN architecture, it is thus desirable to develop intelligent approaches which can automate the process of CNN architecture engineering.

In the literature, to automate the architecture configuration of CNN, there is a growing interest in Neural Architecture Search (NAS), and many optimization and learning approaches, including Bayesian optimization(BO), Reinforcement learning(RL) and gradient-based methods etc., have been proposed to solve this problem. For instance, Bergstra et al. [12] proposed a meta-modeling approach to support automated hyperpatameter optimization, and in [13], Mendoza et al. developed a fully-automatically-tuned neural network which won 2015 official AutoML human expert track competition against human experts. Moreover, Kandasamy et al. [14] proposed a Gaussian process based BO framework for NAS, which can find competitive architectures for MLPs and CNNs efficiently. Further, Zoph and Le. [15] incorporated reinforcement learning into recurrent network to maximize the expected accuracy of the generated architectures. Baker et al. [16] trained an agent, using Q-learning with an ϵ-greedy exploration strategy and experience replay to discover designs with improved performance. Cai et al. [17] also tackled NAS as a sequential decision process with reinforcement learning and used meta-controller to explore the architectures space by reusing the weights of these architectures. However, the success of reinforcement learning methods is mainly based on vast computational resources. More recently, Liu et al. [18] presented DARTS, which can greatly reduce the consumption of computational resource by formulating the NAS in a differentiable manner, which allows gradient descent for efficient search of CNN architectures. In addition, Hundt et al. [19] proposed the sharpDARTS to further accelerate the search

978-1-7281-2547-3/20/$31.00 ©2020 IEEE

process for NAS by incorporating SharpSepConv block and Consine Power Annealing learning rate schedule into DARTS.

Besides the optimization methods discussed above, evolutionary algorithm (EA) also plays an important role to explore the search space for NAS, since it does not require much domain knowledge of the problem when compared to existing algorithms, such as reinforcement learning and gradient-based methods. The early approach which employed EA for NAS is *neuro-evolution of topologies* [20], which only evolved simplified neural network topologies together with weights. With the improvement of hardware, EA can now produce complex architectures for NAS. For instance, Esteban et al. [21] employed genetic alogorithms(GAs) without well-designed crossover operator to search for superior architectures with 250 GPUs against manually designed network. In order to design more available search space, Xie et al. [22] proposed a new encoding method to represent each architecture in a fixed-length binary string and used a genetic algorithm(GA) to find high-quality solutions. Further, Liu et al. [23] casted a novel hierarchical representation in a genetic alogorithm, which obtained state-of-the-art network architecture results. Recently, Esteban et al. [24] designed a tournament selection evolutionary alogorithm with an age property to benefit younger individual in each generation, which obtained highly competitive performance over state-of-the-art models designed by experts. Although these methods obtained encouraging performance, they required large quantities of computation resources. Therefore, Sun et al. [25] proposed a new GA based algorithm for Evolving deep Convolutional Neural Networks (EvoCNN), which can substantially reduce the cost of computation resources. EvoCNN employs a search space of chain-structured neural networks, which is illustrated in Fig.1 (see the left figure in Fig.1), a chain-structured neural network architecture A consists of a sequence of n layers, where the i'th layer L_i only receives its input from layer L_{i-1}, $A = L_n \circ \ldots L_1 \circ L_0$ [26]. Besides, EvoCNN achieved remarkable success on a series of benchmark datasets, such as Fashion benchmark, MBI benchmark, MRD benchmark and MRDBI benchmark. Nevertheless, it is worth noting that there are several drawbacks existed in EvoCNN: 1) the architectures found in EvoCNN are all chain-structured networks, which may cause the degradation problem mentioned in [4]; 2) There may exist some fully connected layers in the final solutions, which are prone to overfitting due to the large quantities of parameters [27]; 3) The performance estimation is still computational expensive.

In order to solve the problems above, in this paper, we propose a memetic algorithm (MA) based neural network MA-Net, which integrates local search into the global search of EvoCNN. Particularly, to solve the first problem, MA-Net employs a well designed local search strategy as illustrated in Fig.1 (see the right figure in Fig.1) to explore the space of a multi-branch network. Further, in order to reduce the influence of overfitting which is caused by excessive number of parameters, MA-Net replaces fully connected layers with convolutional layers to substantially reduce the number of parameters in a network. Finally, for the sake of reducing time consumption, MA-Net compares networks only in the early training process based on the evaluation hypothesis proposed in [28]. To confirm the efficacy of our proposed method for NAS, comprehensive empirical studies are conducted using the datasets considered in EvoCNN, such as Fashion benchmark, MBI benchmark, MRD benchmark, MRDBI benchmark.

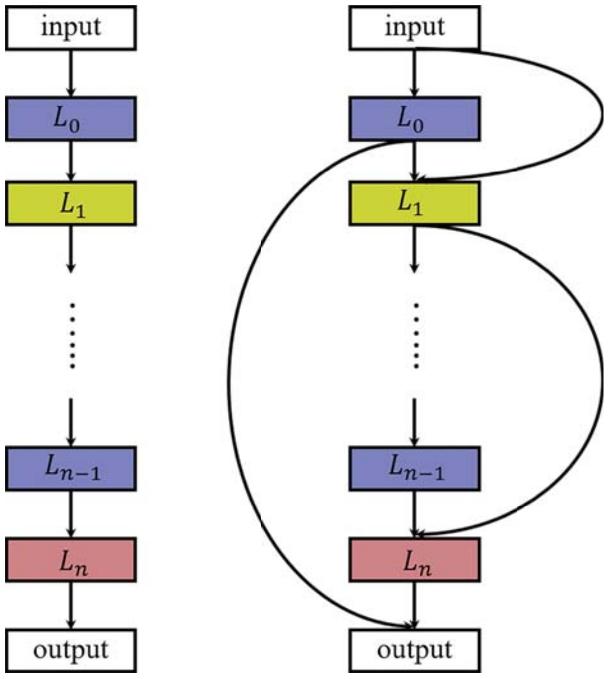

Fig. 1. Two different search spaces. A node in the graphs stands for an operation, which can be a convolution layer or pooling layer, and different colors mean different setting or types. Each connection shows the flow of information between two nodes. Left: chain-structured space used in EvoCNN. Right: multi-branch space used in our proposed Alogorithm.

The rest of this paper is organized as follows. Section II gives a brief introduction of NAS, and a review of related work. Next, the proposed algorithm is presented in Section III. Then, the experimental study and the obtained results are shown in Section IV. Lastly, Section V gives the concluding remarks of this work.

II. Neural Architecture Search

NAS is the process of automating architecture engineering, which can be seen as a subfield of Automated Machine Learning (AutoML) [26]. In NAS, there are generally three core issues: search space, search strategy, and performance estimation strategy. (1) search space contains the set of feasible solutions of a NAS method, and it can be divided into global search space and cell-based search space in image classification [26]. (2) search strategy details how to explore the defined search space, to find well-performing architecture quickly. Existing search strategys mainly contain reinforcement learning (RL) [15], [16], [29], [30], evolutionary algorithm (EA) [21], [22], [24], [25], [31], [32], gradient-based methods [18], [19], multi-objective algorithm (MO) [33]–[35], etc. (3) performance estimation strategy refers to the process of estimating the performance of the found architecture. Typically, the simplest way to estimate an architecture is to train the model until it converges on the training data. However, this process needs a large amount of computational cost. Many approaches have been proposed to simplify this process, including proxy metrics [30], [36], [37], network morphisms [38], [39], one-shot [40]–[43], surrogate model [14], [44], [45], etc.

In this paper, we present MA-Net, which is a NAS method based on memetic algorithm [46], [47] and EvoCNN. Specifically, we mainly focus on the search strategy and design a local search strategy to explore the new designed space of a multi-branch network. Besides, we introduce a faster performance estimation technique based on [28], to accelerate the convergence of the proposed algorithm.

III. Proposed Alogorithm

This section gives the detailed descriptions of our proposed MA-Net. Firstly, we shall give the outline of the proposed algorithm. Next, we present the details of the proposed method with respect to the three aspects as discussed in Section 2, which are search space, search strategy, and performance estimation strategy.

A. Workflow of the proposed MA-Net

Algorithm 1: Framework of the proposed MA-Net

```
1  $P_0 \leftarrow$ Initialize the population with the new designed
     population initialization strategy;
2  Evaluate the fitness of individuals in $P_0$ with the
     proposed evaluation technique;
3  $t \leftarrow 0$;
4  while termination criterion is not satisfied do
5      $S \leftarrow$ Select parent solutions with slack binary
         tournament selection;
6      $Q_t \leftarrow$ Generate offsprings with crossover and
         mutation operators from $S$;
7      Evaluate the fitness of individuals in $Q_t$;
8      Use the well designed local search strategy to
         explore the multi-branch search space for each
         individual in $Q_t$ then evaluate the extra generated
         architectures and update each individual;
9      $P_t + 1 \leftarrow$ Environmental selection from $P_t \cup Q_t$;
10     $t \leftarrow t + 1$;
11 end
12 Select the best individual from $P_t$ and decode it into
     the corresponding convolutional neural network.
```

Alg. 1 shows the workflow of the proposed MA-Net. As depicted, first of all, the population P_0 (line 1) is initialized by a new designed strategy, which will be discussed in section 3.2. Subsequently, the initialized population is evaluated with the proposed evaluation strategy (line 2) that will be discussed in section 3.4. Next, the algorithm generates offsprings via crossover and mutation operators with two randomly selected parent solutions and updates these offsprings with the well designed local search strategy (lines 5-8). It's worth noting that here, the crossover and mutation operators are different from those used in EvoCNN. Due to the outer connections in this study, and in order to change the extra connections, we employ uniform crossover and mutation operators to each bit in the outer connection code. Furthermore, after local search, the extra generated architectures are compared against the initial individual, and the initial one will be replaced if the performance of the extra architectures is better. All the generated individuals during the evolution are evaluated by the new designed performance estimation strategy, which can substantially reduce the time consumption. Finally, once the termination criterion is satisfied, the best individual is selected from the final generation, and then be decoded into the corresponding convolutional neural network (line 12).

B. Search Space

Search space contains a set of solutions found in a NAS approach. In EvoCNN, there are three different building blocks in the search space, the convolutional layer, the pooling layer and the fully conntected layer. However, Due to the fully conntected layer, the architectures found in EvoCNN are complicated with large amounts of weights. Specially, as illustrated in [27], the fully conntected layer is prone to overfitting, thus hampering the generalization ability of the neural network. What's more, the number of weight in the fully conntected layers is extremely larger than that in the convolutional layers. Therefore, the proposed algorithm employs the global average pooling layer to replace the fully conntected layer, and the global pooling layer has the following advantages: (1) there is no weight in this layer, which means this layer can save large quantities of calculation. (2) The global pooling layer could strengthen the correspondences between the feateure maps and categories. Because of this change, the population initialization strategy is significantly different from that in EvoCNN, which is illustrated in Alg. 2.

It can be observed from Alg. 2, first of all, some auxiliary variables are initialized (lines 1-4). Note that, P_0 is an empty array which is used to store the generated convolutional layers and pooling layers (line 1). n_c and n_p are randomly generated numbers of convolutional layer and pooling layer (lines 2-3), respectively. In addition, A_0 denotes a sequence, describing how the convolutional layers and pooling layers stack. Subsequently, to generate a complete architecture sequence, it randomly inserts n_p number of 0 into n_c. Then, according to the modified sequence A_0, the following loop generates a chromosome by orderly stacking a series of convolutional layers and pooling layers (lines 7-15). Finally, we add a global average pooling layer at the top of the chromosome to produce a complete architecture (lines 16-17). By this way, it can generate a population of individuals.

C. Search Strategy

Search strategy is used to explore the search space. The search strategy employed in the proposed alogorithm is an

Algorithm 2: Population Initialization

```
Input: The population size $N$, the maximal number of
       convolutional and pooling layers $N_{cp}$.
Output: Initialized population $P_0$.
1  $P_0 \leftarrow \emptyset$;
2  $n_c \leftarrow$ Randomly generate an integer between [1, $N_{cp}$];
3  $n_p \leftarrow$ Randomly generate an integer between [1,
     $min\{n_c - 1, N - n_c\}$];
4  $A_0 \leftarrow$ Generate an array whose length is $n_c$, and all
     elements in this array are set to 1;
5  Randomly select $n_p$ nodes from $A_0$, except the first
     node. Then insert number zero before all choosed
     nodes in $A_0$;
6  $i \leftarrow 0$;
7  while $i < N$ do
8      if $A_0[i] == 1$ then
9          $l \leftarrow$ Initialize a convolutional layer with
             random settings;
10     else
11         $l \leftarrow$ Initialize a pooling layer with random
             settings;
12     end
13     $P_0 \leftarrow P_0 \cup l$;
14     $i \leftarrow i + 1$;
15 end
16 $part_2 \leftarrow$ Initialize a global average pooling layer;
17 $P_0 \leftarrow P_0 \cup part_2$;
18 Return $P_0$.
```

extension of that in EvoCNN. The major procedure of this search strategy has been shown in the Alg. 1. In particular, as the proposed approach adopts a new designed local search strategy for exploring the multi-branch search space, described in figure 1, this section thus mainly discusses the details about the local search strategy.

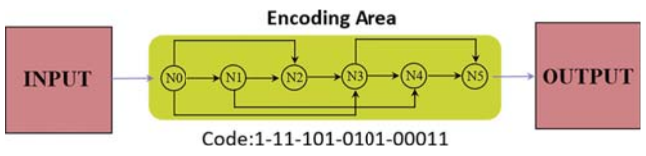

Fig. 2. The encoded sequence. The nodes at both ends are the default input and output, and the middle part is the encoding area. Each circle node represents a convolutional layer and is numbered, node with higher number can receive feature maps from the lower number nodes. Every node employs a short sequence to represent its relationships, and the length of this sequence is same as the number which is assigned to the node, which means the first node won't have a sequence because its number is "0" and it can only be connected to the input. At last, All these short sequences are concatenated by "-", and this long sequence is the final representation of the outer connection.

Particularly, in order to represent the multi-branch search space, we adopt the method proposed in [22] to express the outer connection of the network architecture. As depicted in Figure 2, for simplicity, the input and output are two default nodes in two ends, respectively. Each node representing a convolutional layer is assigned a number in the encoding area. Further, the node with lower number cannot receive information from the node with higher number to avoide cycles. For instance, if the number of one node is i, this node shall has i bits to express its correspondence with other nodes whose numbers are lower than i. As illustrated in Figure 2, the number of $N3$ node is 3, which means $N3$ has 3 bits and there may be three connections between $N3$ and $N0$, $N1$, $N2$. As can be observed, the sequence of $N3$ node is "101", indicating $N3$ can receive information from $N0$ and $N2$. Note that, it is not difficult to change the outer connection for a network architecture. e.g., flip the selected bits. Therefore, using this representation, crossover can be performed by exchanging the sequence of nodes in this representation, while mutation can be conducted via flipping each bit of the chromosome with a certain probability. Moreover, local search is also designed to conduct local exploitation of the neural architecture, which is summarized in Alg. 3.

Algorithm 3: Local Search

```
Input: Individual $I_0$, the number $N$.
Output: An array of individuals $A_{ind}$.
1  if $I_0$ doesn't have outer connection sequence then
2      generate the initial conntection sequence for $I_0$;
3  end
4  $A_c \leftarrow$ Generate all possible conntection sequences for
     $I_0$;
5  $C_0 \leftarrow$ The conntection sequence of $I_0$;
6  Remove the same sequence as $C_0$ from $A_c$;
7  $A_{ind} \leftarrow \emptyset$;
8  $length \leftarrow$ Length of $A_c$;
9  $N' \leftarrow min\{N, length\}$;
10 Random select $N'$ non-repeating sequences from the
     $A_c$ to add to the $A_{ind}$;
11 Return $A_{ind}$
```

As depicted in the Alg. 3, the local search strategy initializes the connection sequence for the input individual I_0, if I_0 does not have connection sequence (lines 2-4). Note that, the initial connection sequence simply represents the chain architecture of I_0. Then, all the possible connection sequences are produced for I_0 (line 5). For instance, suppose individual I_0 has i nodes with convolutional layer, and each ith node owns up to i additional connections, which means I_0 has at most $\frac{(1+i)\times i}{2}$ additional connections. Therefore, there are total $2^{(\frac{(1+i)\times i}{2})}$ different connections choices for I_0, and all these choices are stored in the array A_c. To avoid selecting the same sequence as C_0 from A_c, it is necessary to remove C_0 from A_c (line 7). Finally, to explore the multi-branch space, N' non-repeating sequences are selected from A_c. All the choosed sequence are incorporated into I_0 independently and evaluated with the new designed evaluation strategy as discussed in the following section. Then the best one will be used to represent the outer connection of I_0. After the local search strategy, all the individuals in a population could possess outer conntection sequences.

D. Performance Estimation Strategy

Performance estimation strategy is used to evaluate the solutions of a NAS method. The approach used in EvoCNN is to train the model on a specific benchmark until it converges. However, this way consumes large amounts of computation resources. In order to save large quantities of computation, we reduce the training time to few epochs before it converges, based on the hypothesis proposed in [28]. According to the evaluation hypothesis, we can compare the performance of networks found in the proposed algorithm in the early stage of training process. Therefore, we can simply train these solutions in several or even one epoch to estimate their performance on different datasets.

IV. Experiments

To verify the performance of the proposed MA-Net, comprehensive empirical studies are conducted using the image classification benchmark datasets, which are also used in EvoCNN. In particular, in this section, we first introduce the datasets and experimental settings. Next we present our experimental results and discuss the results on different datasets.

A. Datasets and Experimental Setting

In our experiments, five representative and difficult datasets used in EvoCNN, i.e., the Fashion [48], the MNIST with Background Images (MBI) [49], Random Background (MRB) [49], Rotated Digits (MRD) [49], and with RD plus Background Images (MRDBI) [49] benchmarks, are considered for investigation. Note that, as the Rectangle Images (RI) [49] and the MNIST Basic [49] benchmarks are easy data sets, and most convolutional network approaches can consistently obtain accuracy of 100%, we do not use these two datasets in this study.

These five benchmark datasets can be divided into two categories and the image sizes of them are all 28×28 with single-channel. The first category is the Fashion benchmark, which is a dataset of 10 classes article images, that are shown in Fig.3 (see the top figure in Fig.3). It contains a training set of $60,000$ examples and a test set of $10,000$ examples. Another category contains the variations of the MNIST digits. Due to the simplicity of the basic MNIST benchmark, some different changes are added to the original MNIST benchmark, which are the MBI, MRB, MRD, MRDBI benchmarks, as shown in Fig.3 (see the bottom figure in Fig.3). Moreover, each benchmark only has $12,000$ images in the training set but $50,000$ images in the test set, which means these classification tasks become much more imbalance and difficult than the original MNIST benchmark.

To ensure a fair comparison against EvoCNN, most of the parameter settings are configured according to EvoCNN, as illustrated in Table I. To save computational cost, the population size and maximal generation number are set to be half of that in EvoCNN. If there is no any improvement in the last 8 generations, the evolution will be terminated.

(a) Examples from the Fashion benchmark with 10 classes. From left to right, these images are T-shirt, Trouser, Pullover, Dress, Coat, Sandal, Shirt, Sneaker, Bag, Ankle boot.

(b) Examples from the variations on the MNIST benchmark with 10 classes. From left to right, every two images are from the same benchmark dataset, and they are MBR, MBI, MRD, MRDBI.

Fig. 3. Examples from two kinds of benchmarks. one is the Fashion benchmark, the other are the variations about MNIST benchmark.

Moreover, the maximum length of convolutional layers are 10, since the fully connected layers are replaced by convolutional layers. Typically, as depicted in Alg. 2, the maximum length of pooling layers shall be shorter than the length of convolutional layers, so the maximum length of pooling layers is set to 9 in this study. Moreover, the number of maximum times used to do local search is 2. The training times for an individual are 1 and 4 on Fashion benchmark and the other benchmarks, respectively.

TABLE I
Parameter Settings

Parameters	Settings
Population size and Maximal generation number	50
The distribution index of SBX and Polynomial mutation	20
The maximum lengths of the convolutional layers	10
The maximum lengths of the pooling layers	9
The crossover probability	0.9
The mutation probability	0.1
The maximal local search times	2
The stride in convolutional layers	1
The padding in pooling layers	SAME

We implement the proposed algorithm via Tensorflow [50]. To accelerate the training process and avoid overfitting problem, the BatchNorm is applied to feateure maps after each convolutional layer. Besides, each experiment is executed on a RTX2080ti and no data augmentation approache is used in our method. Lastly, we independently run our method on Fashion benchmark for 5 times, and randomly run on the variations of MNIST benchmark dataset for 1 time.

B. Results and Discussion

All the experimental results compared with EvoCNN on Fashion benchmark dataset and the variations of MNIST benchmarks are summarized in the Table II and Table III, respectively. As presented in Table II, the last two rows show the *best* and *mean* classification errors obtained by the proposed MA-Net, and the other rows show the corresponding results of EvoCNN. Further, to provide more comparisons between the proposed MA-Net and EvoCNN, the additional information of the number of parameters and the computational time on Fashion benchmark dataset are given in the last columns. In

Table III, only the random classification errors about MA-Net are shown in the last row on MRD, MRB, MBI, MRDBI benchmarks, and the first row is the best results obtained by EvoCNN on the corresponding benchmarks. Lastly, all the results are obtained by these two algorithms without any data augmentation preprocessing strategys.

It can be observed from Table II that, by comparing the mean performance, the proposed method MA-Net is better than EvoCNN. The classification error rate of MA-Net is 6.85%, while EvoCNN is 7.28%. In addition, the number of parameters and the time used in MA-Net is only 0.786 million and 1.54 GPU days, which are substantially less than the 6.52 million weights and 4 GPU days of EvoCNN. Although the best classification error rate achieved in EvoCNN is 5.47, slightly smaller than 6.06 obtained in MA-Net, MA-Net only uses 1.24 million parameters and takes just 1.03 GPU days to obtain a competitive architecture. Note that, in spite of the difference between RTX2080ti and GTX1080, the time gap is still significant. Therefore, considering both the *best* and *mean* performance, it is obvious that the performance of MA-Net is more stable and efficient on the Fashion benchmark dataset.

TABLE II
THE CLASSIFICATION ERROR OF THE PROPOSED APPROACHE AGAINST EVOCNN ON FASHION

Classifier	Error	#Parameter	#Time
EvoCNN(best)	5.47	6.68M	4d
EvoCNN(mean)	7.28	6.52M	4d
MA-Net(best)	6.06	1.24M	1.03d
MA-Net(mean)	6.85	0.786M	1.54d

Further, as depicted in Table III, the random performance of MA-Net outperforms the best performance of EvoCNN on all the tested benchmarks - the MRD, MRB, MBI and MRDBI datasets. In particular, on the MRD and MRDBI benchmarks, the classification errors of MA-Net are 3.56% and 15.92%, respectively, which are smaller than 4.53% and 35.03% obtained by EvoCNN. Therefore, the MA-Net also has a better performance on the variations of MNIST.

TABLE III
THE CLASSIFICATION ERROR OF THE PROPOSED APPROACHE AGAINST EVOCNN ON THE MRD, MRB, MBI, MRDBI

Classifier	MRD	MRB	MBI	MRDBI
EvoCNN(best)	5.22	2.80	4.53	35.03
MA-Net(random)	3.33	2.48	3.56	15.92

V. CONCLUSION

In this paper, we have presented MA-Net, which is a new NAS method based on EvoCNN and memetic alogorithm. Experimental studies have verified that this algorithm can find better architectures with fewer parameters and less computational resources when compared to EvoCNN. In the future, we would like to explore more appropriate crossover and mutation operators to accelerate the automation process. In addition, would also like to apply the proposed approach to real world deep learning applications.

VI. ACKNOWLEDGEMENT

This work is partially supported by the National Natural Science Foundation of China (NSFC) under Grant No. 61876025, and the Venture & Innovation Support Program for Chongqing Overseas Returnees under Grant No. cx2018044 and cx2019020.

REFERENCES

[1] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “Imagenet classification with deep convolutional neural networks,” in *Advances in neural information processing systems*, 2012, pp. 1097–1105.
[2] K. Simonyan and A. Zisserman, “Very deep convolutional networks for large-scale image recognition,” *arXiv preprint arXiv:1409.1556*, 2014.
[3] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, “Going deeper with convolutions,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 1–9.
[4] 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.
[5] 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.
[6] S. Ren, K. He, R. Girshick, and J. Sun, “Faster r-cnn: Towards real-time object detection with region proposal networks,” in *Advances in neural information processing systems*, 2015, pp. 91–99.
[7] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, “Ssd: Single shot multibox detector,” in *European conference on computer vision*. Springer, 2016, pp. 21–37.
[8] J. Redmon and A. Farhadi, “Yolov3: An incremental improvement,” *arXiv preprint arXiv:1804.02767*, 2018.
[9] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille, “Semantic image segmentation with deep convolutional nets and fully connected crfs,” *arXiv preprint arXiv:1412.7062*, 2014.
[10] O. Ronneberger, P. Fischer, and T. Brox, “U-net: Convolutional networks for biomedical image segmentation,” in *International Conference on Medical image computing and computer-assisted intervention*. Springer, 2015, pp. 234–241.
[11] V. Badrinarayanan, A. Kendall, and R. Cipolla, “Segnet: A deep convolutional encoder-decoder architecture for image segmentation,” *IEEE transactions on pattern analysis and machine intelligence*, vol. 39, no. 12, pp. 2481–2495, 2017.
[12] J. Bergstra, D. Yamins, and D. D. Cox, “Making a science of model search: Hyperparameter optimization in hundreds of dimensions for vision architectures,” 2013.
[13] H. Mendoza, A. Klein, M. Feurer, J. T. Springenberg, and F. Hutter, “Towards automatically-tuned neural networks,” in *Workshop on Automatic Machine Learning*, 2016, pp. 58–65.
[14] K. Kandasamy, W. Neiswanger, J. Schneider, B. Poczos, and E. P. Xing, “Neural architecture search with bayesian optimisation and optimal transport,” in *Advances in Neural Information Processing Systems*, 2018, pp. 2016–2025.
[15] B. Zoph and Q. V. Le, “Neural architecture search with reinforcement learning,” *arXiv preprint arXiv:1611.01578*, 2016.
[16] B. Baker, O. Gupta, N. Naik, and R. Raskar, “Designing neural network architectures using reinforcement learning,” *arXiv preprint arXiv:1611.02167*, 2016.
[17] H. Cai, T. Chen, W. Zhang, Y. Yu, and J. Wang, “Efficient architecture search by network transformation,” in *Thirty-Second AAAI conference on artificial intelligence*, 2018.
[18] H. Liu, K. Simonyan, and Y. Yang, “Darts: Differentiable architecture search,” *arXiv preprint arXiv:1806.09055*, 2018.
[19] A. Hundt, V. Jain, and G. D. Hager, “sharpdarts: Faster and more accurate differentiable architecture search,” *arXiv preprint arXiv:1903.09900*, 2019.
[20] K. O. Stanley and R. Miikkulainen, “Evolving neural networks through augmenting topologies,” *Evolutionary computation*, vol. 10, no. 2, pp. 99–127, 2002.

[21] 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 *Proceedings of the 34th International Conference on Machine Learning-Volume 70*. JMLR. org, 2017, pp. 2902–2911.

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

[23] H. Liu, K. Simonyan, O. Vinyals, C. Fernando, and K. Kavukcuoglu, "Hierarchical representations for efficient architecture search," *arXiv preprint arXiv:1711.00436*, 2017.

[24] 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.

[25] Y. Sun, B. Xue, M. Zhang, and G. G. Yen, "Evolving deep convolutional neural networks for image classification," *IEEE Transactions on Evolutionary Computation*, 2019.

[26] M. Wistuba, A. Rawat, and T. Pedapati, "A survey on neural architecture search," *CoRR*, vol. abs/1905.01392, 2019. [Online]. Available: http://arxiv.org/abs/1905.01392

[27] M. Lin, Q. Chen, and S. Yan, "Network in network," *arXiv preprint arXiv:1312.4400*, 2013.

[28] X. Zheng, R. Ji, L. Tang, B. Zhang, J. Liu, and Q. Tian, "Multinomial distribution learning for effective neural architecture search," *CoRR*, vol. abs/1905.07529, 2019. [Online]. Available: http://arxiv.org/abs/1905.07529

[29] 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.

[30] 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*, 2018, pp. 8697–8710.

[31] T. Elsken, J.-H. Metzen, and F. Hutter, "Simple and efficient architecture search for convolutional neural networks," *arXiv preprint arXiv:1711.04528*, 2017.

[32] E. Real, A. Aggarwal, Y. Huang, and Q. Le, "Aging evolution for image classifier architecture search," in *AAAI Conference on Artificial Intelligence*, 2019.

[33] T. Elsken, J. H. Metzen, and F. Hutter, "Efficient multi-objective neural architecture search via lamarckian evolution," *arXiv preprint arXiv:1804.09081*, 2018.

[34] 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 *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 517–531.

[35] Z. Lu, I. Whalen, V. Boddeti, Y. Dhebar, K. Deb, E. Goodman, and W. Banzhaf, "Nsga-net: a multi-objective genetic algorithm for neural architecture search," *arXiv preprint arXiv:1810.03522*, 2018.

[36] A. Klein, S. Falkner, S. Bartels, P. Hennig, and F. Hutter, "Fast bayesian optimization of machine learning hyperparameters on large datasets," *arXiv preprint arXiv:1605.07079*, 2016.

[37] P. Chrabaszcz, I. Loshchilov, and F. Hutter, "A downsampled variant of imagenet as an alternative to the cifar datasets," *arXiv preprint arXiv:1707.08819*, 2017.

[38] T. Wei, C. Wang, Y. Rui, and C. W. Chen, "Network morphism," in *International Conference on Machine Learning*, 2016, pp. 564–572.

[39] H. Cai, J. Yang, W. Zhang, S. Han, and Y. Yu, "Path-level network transformation for efficient architecture search," *arXiv preprint arXiv:1806.02639*, 2018.

[40] S. Saxena and J. Verbeek, "Convolutional neural fabrics," in *Advances in Neural Information Processing Systems*, 2016, pp. 4053–4061.

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

[42] H. Pham, M. Y. Guan, B. Zoph, Q. V. Le, and J. Dean, "Efficient neural architecture search via parameter sharing," *arXiv preprint arXiv:1802.03268*, 2018.

[43] G. Bender, "Understanding and simplifying one-shot architecture search," 2019.

[44] 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.

[45] 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.

[46] E. K. Burke, J. P. Newall, and R. F. Weare, "A memetic algorithm for university exam timetabling," in *international conference on the practice and theory of automated timetabling*. Springer, 1995, pp. 241–250.

[47] Y.-S. Ong, M. H. Lim, and X. Chen, "Memetic computation—past, present & future [research frontier]," *IEEE Computational Intelligence Magazine*, vol. 5, no. 2, pp. 24–31, 2010.

[48] H. Xiao, K. Rasul, and R. Vollgraf, "Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms," *arXiv preprint arXiv:1708.07747*, 2017.

[49] H. Larochelle, D. Erhan, A. Courville, J. Bergstra, and Y. Bengio, "An empirical evaluation of deep architectures on problems with many factors of variation," in *Proceedings of the 24th international conference on Machine learning*. ACM, 2007, pp. 473–480.

[50] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin *et al.*, "Tensorflow: Large-scale machine learning on heterogeneous distributed systems," *arXiv preprint arXiv:1603.04467*, 2016.