A New Deep Neural Architecture Search Pipeline for Face Recognition

NING ZHU\(^1\), ZEKUAN YU\(^2\), AND CAIXIA KOU\(^3\)

\(^1\)Huawei Technologies Company, Ltd., Hangzhou 310000, China
\(^2\)Academy for Engineering and Technology, Fudan University, Shanghai 200433, China
\(^3\)School of Sciences, Beijing University of Posts and Telecommunications, Beijing 100876, China

Corresponding authors: Zekuan Yu (yzk@fudan.edu.cn) and Caixia Kou (koucx@lsec.cc.ac.cn)

This work was supported by the National Natural Science Foundation of China (Grant No. 11971073).

ABSTRACT With the widespread popularity of electronic devices, the emergence of biometric technology has brought significant convenience to user authentication compared with the traditional password and mode unlocking. Among many biological characteristics, the face is a universal and irreplaceable feature with simple detection methods and good recognition accuracy. Face recognition is one of the main functions of electronic equipment propaganda. The previous work in this field mainly focused on converting loss function in traditional deep convolution neural networks without changing the network structure. With the development of AutoML, neural architecture search (NAS) has shown remarkable performance in image classification tasks. In this paper, we first propose a new deep neural architecture search pipeline combined with NAS technology and reinforcement learning strategy into face recognition. We quote the framework of NAS, which trains the child and controller networks alternately. At the same time, we optimize NAS by incorporating evaluation latency into rewards of reinforcement learning and utilize the policy gradient algorithm to search the architecture automatically with the cross-entropy loss. The network architectures we searched out have achieved state-of-the-art accuracy in the large-scale face dataset, which achieved 98.77\% top-1 in the MS-Celeb-1M dataset and 99.89\% in LFW dataset with relatively small network size.

INDEX TERMS Neural architecture search, trainable architecture, reinforcement learning, face recognition, large-scale face dataset.

I. INTRODUCTION

Face recognition is extensively utilized in many fields, such as video surveillance, public security, face payment, and smart home. The critical problem in face recognition is how to acquire facial features accurately. According to characteristics, face recognition can be divided into the following two categories: based on shallow features such as SIFT [1], LBP [2], HOG [3] and based on deep convolution neural network(DCNN [4]). The main advantage of deep learning algorithm is that it can be used with a large number of data samples for training and learning the robust face feature representation in different face recognition datasets. This method does not need to design specific features for different types of intra-class differences (such as illuminations, postures, facial expressioin and ages), but can be directly extracted by the deep convolution features. In the training process, bottleneck features representing faces are extracted by DCNN. Then other techniques (such as taking advantage of joint Bayesian or exploiting different loss functions) are used to fine-tune the CNN models. CNN architectures for face recognition have been inspired by those conventional architectures in the ImageNet Large-scale Visual Recognition Challenge (ILSVRC) [5]. Resnet [6] has become the most preferred choice for many target recognition tasks, including face recognition. Many terminals or customers have favored MobileNet’s [7] performance on lightweight devices with limited computing capability. Therefore, the selection of loss function to train CNN methods for these two network frameworks has become the most active research field in face recognition recently. The previous work in this field mainly focused on two directions: converting loss function to improve recognition accuracy in traditional deep convolution neural networks (such as Resnet, Densenet). Another way is to combine the latest loss function with the lightweight system (such as MobileNet and ProxylessNet) to reduce the
size of the network at the minimal expense of accuracy. The development of loss function in advanced face recognition algorithms mainly includes two ideas:

- **Metric Learning**: Contrastive Loss, Triplet Loss [8] and related sampling methods;
- **Margin Based Classification**: Softmax with Center Loss [9], Sphere Face [10], Soft-Margin Loss [11], AM-Softmax (CosFace) [12] and ArcFace [13].

NAS [14] has made remarkable progresses in the past few years. The computationally intensive NAS algorithms based on reinforcement learning or evolutionary algorithms have proved their ability to produce models that transcend traditional network designed by human beings [15]. Other methods, such as DARTS [16], SNAS [17], Proxyless NAS [18], MileNAS [19], and SGAS [20] have also been significantly developed. Although these methods accelerate the search process by reducing search space and changing search strategies, the accuracy is relatively lower. Most NAS methods have trained the image classification tasks, for instance, CIFAR-10 and ImageNet. On these benchmarks, the NAS algorithms have achieved prominent performance, despite the exorbitant computational cost [21].

In this paper, we propose a novel method that applying NAS technology into face recognition, and the neural network structure of the recognition field is customized through enhanced learning algorithm. Considering the influence of network structure size, we combine the latency with the reward of reinforcement learning in the process of neural network search. The goal of this work is to examine whether the NAS method can automatically design better network structures in face recognition.

To prove the effectiveness of this method, we have carried out many experiments. Through the neural network architecture searched by NAS, we achieve 98.77% accuracy in the extensive MS-Celeb-1M [22] dataset and 99.89% accuracy in classical paired LFW [23] dataset. These all meet the industry’s best standards. In addition to achieving outstanding efficiency, the algorithm also has extraordinary performance in terms of network size. The maximum one is only 19.1 M among all the searched networks.

In Section II, we introduce the evolution of loss function in face recognition and the overview of neural architecture search. We describe the detail of NAS algorithm in Section III which is based on reinforcement learning. In Section IV, we actualize various experiments to prove that NAS is superior to traditional face recognition algorithms. Finally, we summarize this paper in Section V and provide some guidance for future research.

### II. RELATED WORKS

The loss functions play an essential role in the gradient updating of neural networks. A proper loss function can significantly improve the efficiency of the neural network and achieve better results. At present, the loss function in face recognition has been substantially developed.

NAS can be divided into many types from two dimensions: search space and search strategy. At the same time, the advantages and disadvantages of NAS are evaluated in terms of acceleration strategy, calculation consumption, parameter numbers, and inference latency. We provide a comprehensive classification of NAS in the following paragraphs.

### A. EVOLUTION OF LOSS FUNCTION

For face recognition tasks, we present five most representative loss functions, namely, Cross-Entropy Loss, Angular-Softmax Loss, Additive Margin Softmax Loss, and ArcFace Loss.

1) **CROSS-ENTROPY LOSS**

The cross-entropy loss is one of the most widely used loss functions in classification scenarios [4]. In face recognition tasks, the cross-entropy loss is an effective method [25] to eliminate outliers, which is expressed as:

$$
L = -\frac{1}{N} \sum_{i=1}^{N} \log \frac{e^{y_i \cdot W^T x_i + b_i}}{\sum_{j=1}^{N} e^{W^T x_i + b_j}}
$$

where $x_i$ is the $i^{th}$ training sample, $N$ is the number of samples, $W_j$ and $W_{j'}$ are the $j^{th}$ and $j'^{th}$ column of $W$, respectively.

2) **ANGULAR-SOFTMAX LOSS**

The Softmax loss is proposed based on softmax. It enables CNN to learn angular resolution features, which is one of the many modifications to softmax function for introducing margin-based learning. It is described as:

$$
L = \frac{1}{N} \sum_{i=1}^{N} - \log \left( \frac{e^{x_i \cdot \psi(\theta_{yi,i})}}{\sum_{j \neq i} e^{x_i \cdot \psi(\theta_{ji,i})}} \right)
$$

where $x_i$ is the $i^{th}$ training sample, $\psi(\theta_{yi,i}) = (-1)^k \cos(m \theta_{yi,i}) - 2k$ for $\theta_{yi,i} \in \left[ \frac{k\pi}{m}, \frac{(k+1)\pi}{m} \right]$, $k \in [0, m-1]$ and $m \geq 1$ is an integer that controls the size of the angular margin.

3) **ADDITIONAL MARGIN SOFTMAX LOSS**

On account of Angular-Softmax Loss, a general function is added to softmax loss for large margin property. The loss function is defined in the following:

$$
L = \frac{1}{N} \sum_{i=1}^{N} \log \frac{e^{s \cdot \cos(\theta_{yi} - m)}}{\sum_{j=1, j \neq i}^{c} e^{s \cdot \cos(\theta_{ji})}}
$$

where $\cos \theta - m$ a general function for large margin property as an additive margin for softmax loss, and the hyper-parameter $s$ is used to scale up the cosine values.

4) **ArcFace LOSS**

Given the above loss functions, a new margin $\cos(\theta + m)$ is proposed, which represents the best geometrical interpretation. The ArcFace Loss function using angular margin is formulated as:

$$
L = \frac{1}{m} \sum_{i=1}^{m} \log \frac{e^{s \cdot \cos(\theta_{yi} + m)}}{\sum_{j=1, j \neq i}^{m} e^{s \cdot \cos(\theta_{ji})}}
$$

91304

VOLUME 8, 2020
where $s$ is the radius of the hypersphere, $m$ is the additive angular margin penalty between $x_i$ and $W_y$, and $\cos(\theta + m)$ is the margin, which makes the class-separations more stringent.

**B. NEURAL ARCHITECTURE SEARCH**

The NAS search space defines the network architecture to ensure the rationality and quality of the sampling model in the search process. For some specific tasks, introducing prior knowledge can reduce the search space, but to a certain extent, it also restricts the generation of the network beyond previous experience. The backbone structure of the NAS network includes chain architecture space, multi-branch architecture space, and block-based search space. Its operator space contains convolution, pooling, residual connection, and other topological structures.

The search strategy of NAS defines what algorithm can quickly and accurately find the optimal configurations of network architecture. According to different search strategies, NAS can be divided into the following categories.

- **Algorithms based on reinforcement learning:** NASNet, BlockQNN [26], ENAS, MnasNet.
- **Algorithms based on evolution learning:** Hierarchical [27], AmoebaNet [28].
- **Continuous differentiable algorithms:** DARTS, NAO [29], DSO [30].
- **Other search algorithms:** SMASH [31], PNAS [32], Auto-Keras [33], Graph hypernet [34], Proxyless NAS, Efficient Multi-Scale Architectures [35]

Since NAS requires a lot of training time and computing capability, the commonly used acceleration strategies are parameter sharing, network morphism, and network pruning. In these NAS algorithms, different tasks and scenarios require various performance evaluation indicators. The indicators used to evaluate the performance of NAS are test accuracy, number of parameters, inference latency, and memory utilization.

**III. PROPOSED METHOD**

We search the MS-Celeb-1M dataset to find network structures and rank them according to the accuracy of network validation. Based on the hypothesis that the accuracy ranking of subnetworks is consistent with that of fixed networks, the top three networks are taken out and trained from scratch. Finally, we test the accuracy of the system on the LFW dataset.

**A. FACE DATASET**

1) TRAINING DATASETS

Among the multitudinous face data, the most representative training data is CASIA-WebFace [36] and MS-Celeb-1M, while the most commonly used test data is LFW (Labeled Faces in the Wild). The CASIA-WebFace contains 4,944,414 face images belonging to 10,575 different individuals. The official MS-Celeb-1M dataset consists of 10M pictures with 100k face identities. Since the original dataset contains much noise, to be fair, we use a high-quality subset of MS-Celeb-1M improved by Arcface authors, which is optimized according to the distance from the image to the center of the class. The dataset consists of 5.8 million images, including 8.5k categories, with dozens to hundreds of faces in each category. Compared with CASIA-WebFace, the data quantities of each label in MS-Celeb-1M are more abundant. So we employ the most generalized and challenging dataset in face recognition (MS-Celeb-1M) as a training set for the NAS.

2) TESTING DATASET

LFW dataset is established to study face recognition in free environments. This collection contains more than 13,000 face images (all from the Internet, not the laboratory environment). Every face is standardized by different names, among which about 1680 people corresponding with more than two faces. According to the standard LFW evaluation scheme, we measure the verification accuracy of 6000 faces. This dataset is widely used to evaluate the performance of the face verification algorithm. Therefore, we use LFW to validate the performance of the network architecture retrieved by NAS.

3) DATA PREPROCESSING

According to the preprocessing technology of face, we use MTCNN [37] to detect five facial markers (eye center, nose tip, and mouth corner) for similarity transformation to align facial images. Every pixel in these images is normalized by subtracting the average of all channels and dividing their standard deviation. At the same time, all the pictures are randomly flipped, padded to 120 × 120 and randomly cut to 112 × 112. The order of all data is shuffled. All of the above preprocessing is executed on both MS-Celeb-1M and LFW datasets. Also, we regularize the training set of MS-Celeb-1M with a 16-size cutout to improve the robustness of the network. We divide the MS-Celeb-1M dataset into training, verification, and test dataset according to the proportion of 8:1:1.

**B. TRAINING DETAILS**

The NAS search space we constructed consists of all possible directed acyclic graphs (DAGs) on L nodes. Fig.1 briefly illustrates the framework of our algorithm.

In the stage of the searching network, we use reinforcement learning algorithm (policy gradient) to guide the controller network to train the optimal child architecture. The goal of reinforcement learning is to find an optimal behavior strategy for the agent to obtain optimal rewards. The policy gradient methods target at modeling and optimizing the policy directly, which rely upon optimizing parameterized policies with respect to the expected return by gradient descent [38]. The controller network and child network are alternately trained. To improve the efficiency of the searching, we share the parameters of the child network. The controller network is consists of a LSTM network which has 100 hidden units.
The main trainable parameters are LSTM’s weights and shared parameters of the child network.

The training process of NAS includes two interweaving stages. When training the child network, the generation strategy of the controller network is steadfast. At the same time, when training the controller network, the weight parameters of the child network are fixed.

1) TRAINING CHILD NETWORK
For each mini-batch of data, LSTM adjusts the policy to sample a network architecture. Momentum is served to optimize the parameters of the child network, by decaying the cross-entropy loss, which is the most commonly used in classification problems. To reduce the size of the search space and the cost of training, thus we draw on the known robust model ENAS and we change the search space. Each node in the child network contains four operations, choosing from $3 \times 3$ or $5 \times 5$ separable convolution, average pooling, and max pooling. To save computational space and reduce the latency of inference, we replace all regular convolutions with depthwise separable convolutions in search space. The convolution blocks definitely as Fig.2(a) are connected in series by a $1 \times 1$ convolution, an $N \times N$ convolution, batch normalization, and RELU, where NAS searches the $N$ and description. At the same time, the pooling block just like Fig.2(b) is relatively straightforward with a $1 \times 1$ convolution added to max or average pooling to limit the channels of operations. Since the parameters are shared, we fix the input and output shapes of each search block. This process goes through $Q$ iterations until all the input pictures are traversed, which represents the completion of an epoch for the child network.

2) TRAINING CONTROLLER NETWORK
After training the epoch of the child network, then we modify its weights and combine different operations and connections to find the optimal architecture. The reward signal updates the gradient of the controller network. Reinforcement learning plays the role of an optimizer. We evaluate child network on each minibatch of the validation set to measure the accuracy, which is substituted into the reward function of reinforcement learning. As shown in the Eq5, we use coefficients to control the trade-off between network accuracy and complexity. Every time the child network infers on the verification set, it obtains the efficiency and the inference latency $L(m)$ of the net. As expected, the inference latency raises with the increasing complexity of network. So we added target latency to the reward function. At the same time, adding arcsin to the multi-objective reward function can provide more positive feedback for the final small accuracy improvement.

$$\maximizem \arcsin(\frac{\text{Accuracy}(m) \times [\frac{\text{Latency}(m)}{\text{Target}}]^q)}{q})$$

where $q$ is the weight factor.

This incentive mechanism has greatly reduced the size of the network. The average verification accuracy and model complexity are balanced in the design of the reward function. To prevent the over-fitting of the model, the controller generates different architectures every step. Finally, we summarize the whole NAS training procedure in Algorithm 1.

3) TRAINING FIXED NETWORK
Data preprocessing includes clipping, flipping, and cutout, with the optimizer function of momentum. According to the accuracy of the validation dataset, we sort the child networks and select the first three structures. All networks are trained from scratch, and the accuracy is verified in the test dataset. The algorithm is exhibited in Algorithm 2.

IV. EXPERIMENT RESULTS
A. NETWORK SETTINGS
At the beginning of the network, we use a $3 \times 3$ convolution to extract features, followed by the batch normalization. As shown in Fig.3(a), the searched skip operations are processed by continuous RELU function, $1 \times 1$ convolution, and batch normalization. These operations not only increase the nonlinearity of the neural network but also control the number of channels in the transmission process. We add a feature...
Algorithm 1 Training NAS With Cross-Entry Loss

1: Input: The training data \( \{x_i, y_i\}_{i=1}^{n} \), the optimize function, the node numbers \( L \), the controller train steps \( S \), the quantity of sampled architecture \( Q \), the target latency \( T \)
2: for each epoch do
3: for minbatch \( B = \{x_i, y_i\}_{i=1}^{B} \) in train_dataset do
4: controller(LSTM) sample architecture = “[arc]”
5: child network update weights \( W \) with “[arc]”
6: end for
7: for step in \( S \) do
8: Accuracy = \( \sum_q^{\text{eval...acc}} \)
9: Latency = \( \sum_q^{\text{eval...lat}} \)
10: Reward = \( \text{arcsin}(\text{Accuracy} \times \left[ \frac{\text{Latency}}{\text{Target}} \right]^q) \)
11: Loss = \( F(\text{reward}) \rightarrow \text{controller update strategy} \)
12: end for
13: end for
14: Output: The network architecture

Algorithm 2 Training Fixed Neural Architecture From Scratch

1: Input: The training data \( \{x_i, y_i\}_{i=1}^{n} \), the fixed architecture [selected arc]
2: controller(LSTM) build network with [selected arc]
3: for each epoch do
4: for minbatch \( B = \{x_i, y_i\}_{i=1}^{B} \) in train_dataset do
5: child network update weights \( W \) with “[selected arc]”
6: end for
7: calculate accuracy on the whole test dataset
8: end for
9: Output: accuracy of test dataset

reduction module between the first three search blocks in the network. In these modules, we halve the height and width, but double the number of channels. We divide each module into two paths according to the number of channels. The first path passes through an average pooling with a stride of 2 and then follows a \( 1 \times 1 \) convolution to fix the number of output channels. In particular, the second path is pre-processed: after a circle of padding around the image, the same size area of the original image in the lower right corner is exercised as a new image. Next up is the same path operation: an average pooling and a \( 1 \times 1 \) convolution. We concatenate the two paths together as a new feature reduction operator. The detailed schematic diagram is exhibited in Fig.3(b). Considering the characteristics of parameter sharing, all skip operations need not only non-linear processing but also feature reduction at similar parallel positions. After the final search blocks, a drop block [39] and a full connection layer are tailgated. In the network architecture, each skip operation is accompanied by the nonlinear module.

B. HYPERPARAMETER SELECTIONS

We train NAS on an 8*V100 machine with a batch size of 128 per GPU. In the processing of training child networks, we adopt the most classical cross-entropy loss function in face recognition. We set the number of search blocks on the child network to 5. To reduce the difference between independent training, we decrease the learning rate by cosine anneal. In addition, we add a function of training epochs [40] to the original cosine annealing, as shown in Eq6, which can converge to the optimal solution faster. To make the neural network better adapt to the tremendous learning rate, it warms up from 0 to 0.1 in the first 20 epochs and then reduces to 0.0001 according to the previous cosine annealing method. We use a weight decay of \( 10^{-4} \), and the optimizing function of the child network is momentum, with its \( \beta \) value being fixed to 0.9.

\[
\begin{align*}
\ell_r &= \ell_r^{\min} + (\ell_r^{\max} - \ell_r^{\min}) \frac{2^{\frac{1}{2}}(1+\cos(\pi \frac{\text{epoch}}{T})) + 1}{2} - 2 \\
\end{align*}
\]

where \( \ell_r^{\min} \) and \( \ell_r^{\max} \) are the minimum and maximum learning rates; \( T \) is the total number of epochs; \( T_{\text{cur}} \) is the current epoch; and \( i \) is the index into a list of these parameters for a sequence of warm restarts in which \( \ell_r^{\max} \) typically decays.

In contrast, the learning rate of the controller network is the strategy that piece-wise decreases from 0.1 to 0.0001 per 20 epochs. At the same time, the controller network uses the stochastic gradient descent (SGD) optimization algorithm. The child and controller networks alternately train 100 epochs, each of which traverses all training and validation data.
C. TRAINING RESULTS

Fig. 4 shows the schematic diagram of the network structure searched by NAS. But the operators in search blocks and the connection of network architectures are different. After the fixed network training is completed, we test the accuracy of the network on the last 35W dataset, which is previously segmented in MS-Celeb-1M. We compared the test results of three networks with other previous studies in this dataset. As shown in Table 1, all the NAS structures exceed previous results. Table 2 shows the test accuracy of different methods on the MS-Celeb-1M dataset.

To further verify the performance of our proposed method, we test it on the LFW dataset, which is the most commonly used in face recognition. We fix the three architectures and their parameters and test them directly on LFW. As expected, NAS-B and NAS-C have achieved remarkable performance except for the slightly worse test results of NAS-A networks. Similar to the regular pattern of fixed network testing, NAS-C surpasses the existing face recognition technology with 99.89% accuracy and achieves the best result.

Although we consider the inference latency in Eq. 5, we set a relatively loose constraint range to improve the network test accuracy.

### Table 1. The training accuracy of different networks with diverse loss functions on the MS-Celeb-1M dataset.

| Network                  | Train Accuracy (%) |
|--------------------------|--------------------|
| Resnet50 & Cross-Entropy | 92.43              |
| Resnet50 & Angular Softmax| 93.33              |
| Resnet50 & Angular Margin Softmax| 93.68              |
| Resnet50 & ArcFace       | 92.34              |
| Resnet50 & Marginal Loss | 92.77              |
| MobileNet v1 & Cross-Entropy | 93.91              |
| MobileNet v1 & Angular Softmax| 93.45              |
| MobileNet v1 & Angular Margin Softmax| 94.10              |
| MobileNet v1 & ArcFace   | 93.44              |
| NAS-A & Cross-Entropy   | 92.34              |
| NAS-B & Cross-Entropy   | 98.69              |
| NAS-C & Cross-Entropy   | 98.77              |

### Table 2. The final test accuracy of the top three fixed networks searched by NAS on the MS-Celeb-1M dataset. The test accuracy of NAS-A reaches 98.32%, NAS-B achieves 98.69%, and NAS-C attains 98.77%.

| Method            | Test Accuracy on MS-Celeb-1M (%) |
|-------------------|----------------------------------|
| NAS-A             | 98.32                            |
| NAS-B             | 98.69                            |
| NAS-C             | 98.77                            |

### Table 3. The testing accuracy of the top three fixed networks searched by NAS on the LFW dataset. The testing accuracy of NAS-A reaches 99.80%, NAS-B achieves 99.84% and NAS-C attains 99.89%.

| Method           | Test Accuracy on LFW (%) |
|------------------|--------------------------|
| Deep Face [42]   | 97.35                    |
| DeepFR [24]      | 98.95                    |
| Center Loss      | 99.28                    |
| LightCNN [43]    | 99.33                    |
| SphereFace       | 99.42                    |
| Marginal Loss    | 99.48                    |
| Range Loss       | 99.52                    |
| DeepID [46]      | 99.52                    |
| MobileFaceNet [47]| 99.55                 |
| FaceNet          | 99.63                    |
| CosFace [48]     | 99.73                    |
| Resnet50 & Cross-Entropy | 97.42                |
| Resnet50 & Angular Softmax| 99.10                |
| Resnet50 & Angular Margin Softmax| 99.34               |
| Resnet50 & ArcFace Loss | 99.35              |
| Resnet100 & ArcFace Loss | 99.83              |
| MobileNet & Cross-Entropy | 96.78                |
| MobileNet & Angular Softmax| 97.83                |
| MobileNet & Angular Margin Softmax| 98.46               |
| MobileNet & ArcFace Loss | 99.01              |
| NAS-A            | 99.80                    |
| NAS-B            | 99.84                    |
| NAS-C            | 99.89                    |
accuracy as much as possible. When the network is small, the speed of precision reduction is much slower than that of parameter reduction. However, the complexity of the network structure is not strongly related to the test accuracy. Among the three network structures previously searched, NAS-B has the largest size, reaching 19.1M, but the highest efficiency is NAS-C, which has only 16M. Despite this, it is still better than traditional Resnet. As the structure of the final neural network has been fixed, the training time mainly depends on the size of the network. Thanks to the addition of latency in the reward of reinforcement learning, the network structure found by NAS is not so large. Therefore, compared with Resnet and Mobilenet, the training time of our fixed architecture is on the same order of magnitude. This experimental result helps select the appropriate architecture, which can obtain relatively small networks under the premise of excellent performance.

V. CONCLUSIONS AND FUTURE WORK

In the solution of the convolutional neural network, the efficiency of the model is mainly determined by the architecture of backbone networks. In this paper, we change the traditional method by using the classical network as a backbone and modifying loss in face recognition with NAS. The neural network we searched surpasses all previous results on the MS-Celeb-1M dataset with 98.77% accuracy. At the same time, we apply the search results directly to the LFWin dataset, and the test accuracy also reaches 99.89% accuracy. The performance on these two famous face datasets proves the universality of our method. On the premise of guaranteeing the accuracy, we reduce the size of the searched network structure as much as possible.

In the future, we would like to improve the efficiency of the NAS search and design a more general search space. What’s more, we will test the model with several face recognition datasets, considering low and high resolutions, face positions, and light conditions in the future work. We believe that the combination of NAS strategy and various loss functions is also a viable research direction.

REFERENCES

[1] D. G. Lowe, “Object recognition from local scale-invariant features,” in Proc. 7th IEEE Int. Conf. Comput. Vis., Corfu, Greece, Sep. 1999, pp. 1150–1157.
[2] C. Shan, S. Gong, and P. W. McOwan, “Facial expression recognition based on local binary patterns: A comprehensive study,” Image Vis. Comput., vol. 27, no. 6, pp. 803–816, May 2009.
[3] F. Xu and Z. Wang, “A facial expression recognition method based on cubic spline interpolation and HOG features,” in Proc. IEEE Int. Conf. Robot. Biomimetics (ROBIO), Dec. 2017, pp. 884–888.
[4] A. Krizhevsky, I. Sutskever, and G. Hinton, “ImageNet classification with deep convolutional neural networks,” in Proc. Adv. Neural Inf. Process. Syst. (NIPS), 2012, pp. 1097–1105.
[5] O. Russakovsky, J. Deng, and H. Su, “ImageNet large scale visual recognition challenge,” Int. J. Comput. Vis., vol. 115, no. 3, pp. 211–252, 2014.
[6] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in Proc. CVPR, 2016, pp. 770–778.
[7] G. A. Howard, M. Zhu, and B. Chen, “MobileNets: Efficient convolutional neural networks for mobile vision applications,” Apr. 2017, arXiv:1704.04861. [Online]. Available: https://arxiv.org/abs/1704.04861
[8] F. Schroff, D. Kalenichenko, and J. Philbin, “FaceNet: A unified embedding for face recognition and clustering,” in Proc. CVPR, 2015, pp. 815–823.
[9] Y. Wen, K. Zhang, Z. Li, and Y. Qiao, “A discriminative feature learning approach for deep face recognition,” in Proc. ECCV, 2016, pp. 499–515.
[10] W. Liu, Y. Wen, Z. Yu, M. Li, B. Raj, and L. Song, “SphereFace: Deep hypersphere embedding for face recognition,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR), vol. 1, Jul. 2017, pp. 212–220.
[11] X. Liang, X. Wang, Z. Lei, S. Liao, and S. Z. Li, “Soft-margin softmax for deep classification,” in Proc. ICONIP, 2017, pp. 413–421.
[12] F. Wang, J. Cheng, W. Liu, and H. Liu, “Additive margin softmax for face verification,” IEEE Signal Process. Lett., vol. 25, no. 7, pp. 926–930, Jul. 2018.
[13] J. Deng, J. Guo, N. Xue, and S. Zafeiriou, “ArcFace: Additive angular margin loss for deep face recognition,” in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR), Jun. 2019, pp. 4690–4699.
[14] B. Zoph and Q. V. Le, “Neural architecture search with reinforcement learning,” in Proc. Int. Conf. Learn. Representations (ICLR), 2017, pp. 1–16.
[15] H. Pham, M. Y. Guan, B. Zoph, Q. V. Le, and J. Dean, “Efficient neural architecture search via parameter sharing,” in Proc. ICLR, 2018, pp. 1–11.
[16] H. Liu, K. Simonyan, and Y. Yang, “Darts: Differentiable architecture search,” in Proc. ICLR, 2019, pp. 1–13.
[17] S. Xie, H. Zheng, C. Liu, and L. Lin, “SNAS: Stochastic neural architecture search,” in Proc. ICLR, 2019, pp. 1–17.
[18] H. Cai, L. Zhu, and S. Han, “ProxylessNAS: Direct neural architecture search on target task and hardware,” in Proc. Int. Conf. Learn. Representations, 2019, pp. 1–13.
[19] H. Cai, L. Zhu, and S. Han, “MiLeNAS: Efficient neural architecture search via mixed-level reformulation,” in Proc. CVPR, 2020, pp. 1–13.
[20] G. Li, G. Qian, C. I. Delgadillo, M. Müller, A. Thabet, and A. Thabet, B. Ghanem, “SGAS: Sequential greedy architecture search,” in Proc. CVPR, 2020, pp. 1–18.
[21] M. Tan, B. Chen, R. Pang, V. Vasudevan, and Q. V. Le, “MnasNet: Platform-aware neural architecture search for mobile,” in Proc. CVPR, 2019, pp. 2820–2828.
[22] Y. Guo, L. Zhang, Y. Hu, X. He, and J. Gao, “NAS-C: A dataset and benchmark for large-scale face recognition,” in Computer Vision—ECCV (Lecture Notes in Computer Science), vol. 9907, B. Leibe, J. Matas, N. Sebe, and M. Welling, Eds. Cham, Switzerland: Springer, 2016, pp. 87–102.
[23] G. B. Huang, M. Mattar, T. Berg, and E. Learned-Miller, “Labeled faces in the wild: A database for studying face recognition in unconstrained environments,” in Proc. Workshop Faces Real-Life Images, Detection, Alignment, Recognit., 2008, pp. 1–15.
[24] O. M. Parkhi, A. Vedaldi, and A. Zisserman, “Deep face recognition,” in Proc. Brit. Mach. Vis. Conf., 2015, vol. 1, no. 3, pp. 1–12.
[25] Y. Sun, Y. Chen, X. Wang, and X. Tang, “Deep learning face representation by joint identification-verification,” in Proc. Adv. Neural Inf. Process. Syst., 2014, 1988–1996.
[26] Z. Zhong, Z. Yang, B. Deng, J. Yan, W. Wu, J. Shao, and C.-L. Liu, “BlockQN: Efficient block-wise neural network architecture generation,” IEEE Trans. Pattern Anal. Mach. Intell., early access, Jan. 23, 2020, doi: 10.1109/TPAMI.2020.2969193.
[27] C. Liu, L.-C. Chen, and F. Schroff, H. Adam, W. Hua, A. L. Yuille, and L. Fei-Fei, “Auto-deeplab: Hierarchical neural architecture search for semantic image segmentation,” in Proc. CVPR, 2019, pp. 82–92.
[28] E. Real, A. Aggarwal, Y. Duan, and Q.-V. Le, “Reranking evolution for image classifier architecture search,” in Proc. AAAI, 2019, pp. 4780–4789.
[29] R. Luo, F. Tian, T. Qin, E.-H. Chen, and T.-Y. Liu, “Neural architecture optimization,” in Proc. Adv. Neural Inf. Process. Syst., 2018, pp. 7816–7827.
[30] X. Zhang, Z. Huang, and N. Wang, “You only search once: Single shot neural architecture search via direct sparse optimization,” in Proc. ICLR, 2019, pp. 1–11.
[31] A. Brock, T. Lim, J. M. Ritchie, and N. Weston, “Smash: One-shot model architecture search through hypernetworks,” in Proc. ICLR, vol. 10, 2018, pp. 1–21.
[32] C. Liu, B. Zoph, J. Shlens, and W. Hua, “Progressive neural architecture search,” in Proc. ECCV, 2018, pp. 1–34.
[33] H. Jin, Q. Song, and X. Hu, “Auto-Keras: An efficient neural architecture search system,” in Proc. Int. Conf. Knowl. Discovery Data Mining, 2019, pp. 1946–1956.
[34] C. Zhang, M. Ren, and R. Urtasun, “Graph hypernetworks for neural architecture search,” in Proc. ICLR, 2019, pp. 1–17.

[35] L. Chen, D. M. Collins, and Y. Zhu, “Searching for efficient multi-scale architectures for dense image prediction,” in Proc. NIPS, 2018, pp. 8699–8710.

[36] D. Yi, Z. Lei, S. Liao, and Z. S. Li, “Learning face representation from scratch,” Nov. 2014, arXiv:1411.7923. [Online]. Available: https://arxiv.org/abs/1411.7923

[37] K. Zhang, Z. Zhang, Z. Li, and Y. Qiao, “Joint face detection and alignment using multitasking cascaded convolutional networks,” IEEE Signal Process. Lett., vol. 23, no. 10, pp. 1499–1503, Oct. 2016.

[38] R. S. Sutton, D. A. McAllester, S. P. Singh, and Y. Mansour, “Policy gradient methods for reinforcement learning with function approximation,” in Proc. NIPS, 1999, pp. 1057–1063.

[39] G. Ghiasi, T.-Y. Lin, and V. Q. Le, “DropBlock: A regularization method for convolutional networks,” in Proc. NIPS, 2018, pp. 10727–10737.

[40] A. Hundt, V. Jain, and G. D. Hager, “SharpDARTS: Faster and more accurate differentiable architecture search,” Mar. 2019, arXiv:1903.09900. [Online]. Available: https://arxiv.org/abs/1903.09900

[41] Y. Srivastava, V. Muruli, and A. R. Dubey, “A performance comparison of loss functions for deep face recognition,” in Proc. NCVRIPG, 2019, pp. 1–10.

[42] Y. Taigman, M. Yang, M. Ranzato, and L. Wolf, “DeepFace: Closing the gap to human-level performance in face verification,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit. CVPR, Jun. 2014, pp. 1701–1708.

[43] X. Wu, R. He, and Z. Sun, “A lightened CNN for deep face representation,” Nov. 2015, arXiv:1511.02683. [Online]. Available: https://arxiv.org/abs/1511.02683

[44] J. Deng, Y. Zhou, and S. Zafeiriou, “Marginal loss for deep face recognition,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit. Workshops (CVPRW), Jul. 2017, pp. 60–68.

[45] X. Zhang, Z. Fang, Y. Wen, Z. Li, and Y. Qiao, “Range loss for deep face recognition with long-tailed training data,” in Proc. IEEE Int. Conf. Comput. Vis. (ICCV), Oct. 2017, pp. 5409–5418.

[46] Y. Sun, D. Liang, X. Wang, and X. Tang, “Deepid3: Face recognition with very deep neural networks,” in CoRR, vol. 1502.00873, 2015.

[47] S. Chen, Y. Liu, X. Gao, and Z. Han, “MobileFaceNets: Efficient CNNs for accurate real-time face verification on mobile devices,” in Biometric Recognition (Lecture Notes in Computer Science), vol. 10996, J. Zhou et al., Eds. Cham, Switzerland: Springer, 2018.

[48] H. Wang, Y. Wang, Z. Zhou, X. Ji, D. Gong, J. Zhou, Z. Li, and W. Liu, “CosFace: Large margin cosine loss for deep face recognition,” in Proc. CVPR, Jun. 2018, pp. 5265–5274.

NING ZHU was born in Jiangsu, China. He received the B.Sc. degree in materials science and engineering from Beihang University, Beijing, China, and the M.Sc. degree in radiation protection from the China Academy of Engineering Physics, Beijing. From 2015 to 2018, he was an Image Engineer at Zhejiang Dahua Technology Company, Ltd. Since 2018, he has been an AI Algorithm Engineer at Huawei Technologies Company, Ltd. His research interests include computer vision, AutoML, and reinforcement learning.

ZEKUAN YU received the B.S. degree in electrical engineering and automation from the China University of Mining and Technology, in 2014, and the Ph.D. degree from Peking University, China, in 2020. Since 2020, he joined the Academy for Engineering and Technology, Fudan University. His current research interests include medical image analysis and computer aided diagnosis, deep learning, and medical robot applications.

CAIXIA KOU received the B.Sc. degree from the Department of Mathematics and Applied Mathematics, Beijing Normal University, China, in 2006, and the Ph.D. degree from the State Key Laboratory of Scientific and Engineering Computing (LSEC), Chinese Academy of Sciences (CAS), Beijing, China, in 2011. Since 2011, she joined the School of Sciences, Beijing University of Posts and Telecommunications. Her major field is optimization algorithms and applications. She has proposed several efficient practical optimization algorithms and released the corresponding software packages. Her current research interests include large-scale optimization methods and machine learning.