Evolutionary Neural Architecture Search for Facial Expression Recognition

Shuchao Deng, Zeqiong Lv, *Student Member, IEEE*, Edgar Galván, *Member, IEEE*, and Yanan Sun, *Member, IEEE*

***Abstract*—Facial expression is one of the most powerful, natural, and universal signals for human beings to express emotional states and intentions. There are many applications for facial expression recognition (FER) in human society such as healthcare. Thus, the importance of correct and innovative FER approaches in Artificial Intelligence is evident. However, commonly used methods suffer from a lack of classification generalization in FER. To tackle this problem, we propose a generic facial expression recognition network based on evolutionary neural architecture search, called ENAS-FERNet, which can automatically evolve neural network architectures using both laboratory-controlled and in-the-wild FER datasets. The experiments of ENAS-FERNet were carefully designed and compared with state-of-the-art (SOTA) methods in the case of training from scratch. In addition, we validated the interference resistance of ENAS-FERNet on the synthetic noisy FER dataset and analyzed the time consumption of ENAS-FERNet. Comprehensive experimental analysis and results show that the proposed ENAS-FERNet method achieves the most well-known results on the CK+, Affect-Net, and RAF-DB (10%) datasets, as well as competitive results on the JAFFE, RAF-DB, and RAF-DB (20%) datasets. The results of these experiments show that our ENAS-FERNet has good classification generalization capabilities on these challenging datasets.**

***Index Terms*—Facial expression recognition, evolutionary neural architecture search, neural network architectures.**

I. Introduction

FACIAL expression recognition (FER) has a variety of applications in human society, such as medical care, human-computer interaction, communication, automotive, and robotics manufacturing [1], [2], to mention some. FER has recently attracted increasing attention in the research community and it is not surprising to see a large number of scientific papers in this area, many of which are focused on proposing methods that perform well on FER tasks [3]. According to whether the design of neural network architectures can be automated, common FER methods include manually designed neural network architectures using human expertise [3], and automatically designed neural network architectures through neural architecture search (NAS) [4]. The former is very common for FER. Manually designed neural network architectures in the early stages mainly used hand-crafted feature extraction or shallow learning techniques (e.g., local binary patterns [5]). The most important properties of the local binary-based method are its tolerance to light variations and the simplicity of its calculations, but with the advent of various challenging datasets such as RAF-DB [6] and Affect-Net [7], the method has struggled to be effective. Nowadays, with the rapid increase of data, deep learning (DL) [8] algorithms, a subset of machine learning algorithms, are inspired by deep hierarchical structures of human perception as well as production systems. These algorithms have achieved extraordinary results in different exciting areas including computer vision, speech recognition, board games, and video games, to mention a few examples. Multiple DL architectures have been proposed in the specialized literature, where convolutional neural networks (CNNs) [9] are the most well-known networks thanks to their wide applicability in Euclidean data problems. These CNNs have become the standard architectures for FER tasks in multiple scientific works such as Deep-Emotion [3], Emotion-FAN [10], RAN [11], and SCN [12] in laboratory-controlled and in-the-wild datasets, thanks to outperforming other non-CNN techniques. While these manually designed network architectures have yielded interesting results, finding a well-performing architecture is often a very tedious and error-prone process [13]. Moreover, as the related tasks change, the corresponding network architecture should very often change too. It is obvious that there is a strong need to design an automatic network architecture method for FER.

NAS [4] is an area of growing interest as demonstrated by the large number of scientific works published in recent years [14]. Broadly speaking these works can be classified into three different categories: reinforcement learning (RL)-based [4] methods, gradient-based [15] methods, and evolution-based methods [16]. Other methods outside of these three categories have also been proposed in the specialized literature including Monte Carlo-based simulations [17], random search [18], and sequential model-based optimization (SMBO) [19]. The RL-based method regards architecture generation as a process in which an intelligent agent selects an action and tests the network performance on a test set to obtain reward values that guide the generation of the architecture. There are a lot of works based on RL emerging

Manuscript received 19 January 2023; revised 4 April 2023; accepted 18 May 2023. Date of publication 10 July 2023; date of current version 25 September 2023. This work was supported in part by the National Natural Science Foundation of China under Grant 62276175, in part by Zhejiang Lab under Grant 2022PG0AB02, and in part by Key R&D Projects in Sichuan Province under Grant 2023YFG0031. *(Corresponding author: Yanan Sun.)*

Shuchao Deng, Zeqiong Lv, and Yanan Sun are with the College of Computer Science, Sichuan University, Chengdu 610065, China (e-mail: shuchao@stu.scu.edu.cn; zq_lv@stu.scu.edu.cn; ysun@scu.edu.cn).

Edgar Galván is with the Naturally Inspired Computation Research Group, Department of Computer Science, Maynooth University, W23 F2K8 Maynooth, Ireland (e-mail: edgar.galvan@mu.ie).

Digital Object Identifier 10.1109/TETCI.2023.3289974

in many kinds of literature such as NAS-RL [4], MetaQNN [20], EAS [21], Block-QNN-S [22], and NASNet [23]. Among them, NAS-RL and MetaQNN are widely recognized as pioneers in the field of NAS. Specifically, NAS-RL employs a recurrent network for generating model descriptions of neural networks and uses reinforcement learning for training this RNN, which maximizes the expected accuracy of the generated architectures on the specified validation set. MetaQNN automatically generates high-performing neural network architectures for a given learning task based on reinforcement learning. However, both NAS-RL and MetaQNN have the problem of being time-consuming [14]. To reduce the search cost, EAS explores the architecture space based on the current network and reuses its weights. At the same time, BlockQNN is proposed for the same purpose, which automatically builds high-performance networks using the Q-Learning paradigm with an epsilon-greedy exploration strategy. From the design of search space to reduce the search cost, NASNet designs a transferable search space, which can find a building block on the small dataset and then transfer the found block to the large dataset without researching. Although these RL-based methods can reduce the cost to some extent, they are still inefficient because they regard NAS as a black-box optimization problem in a discrete search strategy.

To be more efficient, the gradient-based method is developed, and a number of representative works have emerged recently. For example, the well-known DARTS [24] transforms the space of discrete neural architectures into continuous forms and further uses gradient optimization techniques to search for neural architectures. DAS [25] is similar to DARTS but DAS focuses more on the hyperparameters of the searched convolutional layers. For sufficient stability during the process of search, P-DARTS [26] is developed by using progressive search for increasing the depth of the network architecture and proposing regularization of the search space. To reduce memory usage and improve the search efficiency of DARTS, Xu et al. [27] propose PC-DARTS, which applies channel sampling for replacing the convolution operation on all channels in DARTS. Although the gradient-based approach can efficiently address the large time-consuming drawbacks of RL-based methods, this approach relies heavily on human intervention. To achieve the purpose of reducing human intervention as much as possible, researchers propose evolution-based methods inspired by nature. For example, Large-scale Evolution [28] is seen as a pioneer in the use of evolutionary strategies for NAS and evolves an optimal architecture automatically from the simplest single-layer network. Subsequently, more and more evolutionary algorithms for NAS are proposed from reducing the search cost and enriching the search space. Among them, GeNet [29] proposes a new encoding strategy for alleviating the problem of large search space in Large-scale Evolution, which represents the network architecture as a fixed-length string. To achieve more flexible encoding, Masanori et al. [30] use Cartesian genetic programming (CGP) to automatically construct CNN architectures as a variable-length string for image classification. Furthermore, to make the architecture more diverse and efficient, AmoebaNet [31] proposes aging evolution, a variant of tournament selection that facilitated the retention of individuals with potential. AmoebaNet has the ability to explore a diverse search space by introducing age evolution which ensures diversity and merit in the evolutionary process.

Although these NAS algorithms perform better than those hand-crafted methods, these works still have some limitations. First, the RL-based methods require high computational power. For example, to obtain a well-performing network architecture on the CIFAR-10 dataset, NAS-RL [4] and NASNet [23] took 22,400 and 2,000 GPU Days, respectively. Other methods such as MetaQNN [20] and Block-QNN [22] consumed about 100 GPU Days, and even ENA [21], a specific optimized method, required at least 10 GPU Days. As mentioned above, RL-based methods are too time-consuming to be used on challenging FER datasets. Second, the gradient-based methods require a priori knowledge. For example, the DARTS-like methods such as DAS [25], P-DARTS [26], and PC-DARTS [27] choose to optimize a predefined super network directly and the best subnetwork can be decoupled from the super-network according to the learned hybrid operation weights. However, predetermining an ideal supernetwork is a difficult process and relies heavily on prior knowledge [14], [24], [32]. Therefore, the gradient-based methods are difficult to achieve a completely automated search on diverse and challenging FER datasets, i.e., their classification generalization capability is weak. Finally, evolution-based methods can achieve completely automated search network architectures, but existing approaches still have the limitation of requiring a large amount of computation. For example, the Large-scale Evolution [28] and AmoebaNet [31] mentioned above spend 2,600 and 3,150 GPU Days on the CIFAR-10 dataset, respectively. Although other evolution-based methods such as GeNet [29] and CGP [30] spent 17 and 14.9 GPU Days respectively on the CIFAR-10 dataset, they are still time-consuming and GeNet as mentioned above has the limitation of fixed-length encoding. To reduce the high time consumption, improve the classification generalization capability and improve flexibility of variable-length encoding, we propose a generic automated Facial Expression Recognition Network based on Evolutionary Neural Architecture Search called ENAS-FERNet. Specifically, ENAS-FERNet can achieve a completely and generically automated search based on GAs. And ENAS-FERNet can achieve efficient search by designing the basic unit module combined with the proposed variable-length encoding strategy. Therefore, ENAS-FERNet can automatically evolve based on different datasets and ultimately return a network architecture with good performance. The main contribution of this work is addressing these issues that impede further progress of NAS on FER research. Specifically, the contributions of this work are as follows:

- We propose a generic network architecture model for automatic FER called ENAS-FERNet, which can automatically find a well-performing architecture for a given dataset and avoid the tedious and error-prone process involved in manual methods. To the best of our knowledge, this is the first automated facial expression recognition network designed in combination with GAs.
- ENAS-FERNet is able to encode neural network architectures as individuals in a population and to automatically evolve them for different datasets to obtain a well-performing network architecture, thus solving the

problem of poor classification generalization that exists in current automated methods.

- ENAS-FERNet contains several features. First, we propose the variable-length encoding strategy to improve the efficiency of variable-length encoding in traditional GAs. Next, skip connection is incorporated into the proposed algorithm to handle complex data. Finally, a global cache system is set up to search more efficiently.
- The experimental results show that ENAS-FERNet has good classification generalization capabilities and solves the time-consuming problems of existing automated methods. Specifically, it achieves the best-known results on the CK+, Affect-Net, and RAF-DB (10%) datasets, and competitive results on the JAFFE, RAF-DB, and RAF-DB (20%) datasets in the case of training from scratch.

The remainder of this article is organized as follows. Background on NAS is provided in Section II. In Section III, we provide a detailed description of the related work on FER, after which we specify the proposed network model ENAS-FERNet in Section IV. The experiments and related analysis are then discussed in Section V. Finally, the conclusion and future work are given in Section VI.

II. Background on NAS

A. Neural Architecture Search

Machine learning and deep learning are now being used in an increasing number of fields such as computer vision, healthcare, communication, automotive, and robotics. This has led to manual and automated methods for the correct configuration of these networks. Google's proposal of NAS [4] caused a boom in the research community. Since then, NAS has attracted an increasing number of researchers due to its ability to automatically search for a well-performing network [33]. There are three main parts in NAS: search space [21], search strategy [31], and performance estimation [20].

Search space: The search space defines the search area in which architectures can be searched or represented. The size of the search space affects the speed of the search. Specifically, a large search space can lead to a search process that takes too much time or even fails to effectively search for network architectures with good performance, and *vice versa*, a small search space can lead to a very limited search for network architectures that may not contain network architectures with good performance. Combining a priori knowledge about network architectures that are well suited for a particular task can reduce the size of the search space, and thus, simplifying the search. This, however, introduces human bias [34], which may prevent finding novel architectures that are beyond the reach of current human knowledge. With the development of NAS, the construction of search spaces is divided into two main categories, namely chain-based search spaces [34] and cell-based search spaces [31].

Search strategy: A search strategy is a method used to search for how to efficiently search in the exponential large or even unbounded search space. It weighs up the trade-off between how quickly to explore and how efficiently to exploit the search space. Since, on the one hand, it is desirable to quickly find well-performing architectures and, on the other hand, avoid premature convergence by getting stuck in local optima [34]. In general, search strategies can be divided into reinforcement learning-based search strategies [4], gradient-based search strategies [24] and evolutionary computation-based search strategies [35]. Each of these three search strategies has its own performance characteristics. In terms of search speed, the reinforcement learning-based search strategy requires a high time cost, while the gradient descent-based search strategy is fast, and the evolutionary-based search strategy lies between the two.

Performance estimation: The goal of NAS is usually to find network architectures that achieve high (classification) performance on data. Performance estimation in general represents the evaluation of the performance of network architectures. For the field of computer vision, the most common approach is to validate the searched network architecture on a specific dataset. Unfortunately, however, this approach is computationally expensive [31], while somewhat hindering the network architectures that can be searched for. As a result, the ability to effectively reduce the search cost is a very worthwhile area of research. Recently, many researchers have focused on developing methods to reduce the cost of performance estimation [36], such as the weight sharing mechanism [24], early stopping mechanism [37], and performance predictor [38].

Although existing automated methods have achieved good results compared to manually designed neural network architectures, they still have many limitations. On the one hand, DARTS and Auto-FERNet require a lot of a priori knowledge, which is often difficult to obtain, resulting in a lack of classification generalization capability due to the lack of a priori knowledge on different datasets, which can lead to good results on some datasets, but poor results on others. On the other hand, ConvGP [39] and CGP are approaches based on genetic programming from a fully automated perspective, which can fully automate the search for neural network architectures for different datasets, but these approaches have the limitation of being time-consuming.

B. Genetic Algorithm

In this work, we propose a generic automated FER network named ENAS-FERNet, which aims to solve the problems of poor classification generalization and high time required in existing automated methods. For the FER task, we use a genetic algorithm (GA) to automatically search well-performing network architectures. This evolutionary algorithm was introduced by Holland [40] in the 1970 s and was highly popularized by Goldberg [41]. This was due to the fact of achieving extraordinary results as well as reaching multiple research communities, including the machine learning and the neural networks communities. GAs was frequently described as function optimizers, but now the tendency is to consider GAs as search algorithms able to find near-optimal solutions. Multiple forms of GAs have been proposed in the specialized literature. The bitstring fixed-length representation is one of the most predominant encodings used in GAs. Crossover, as the main genetic operator, and mutation as

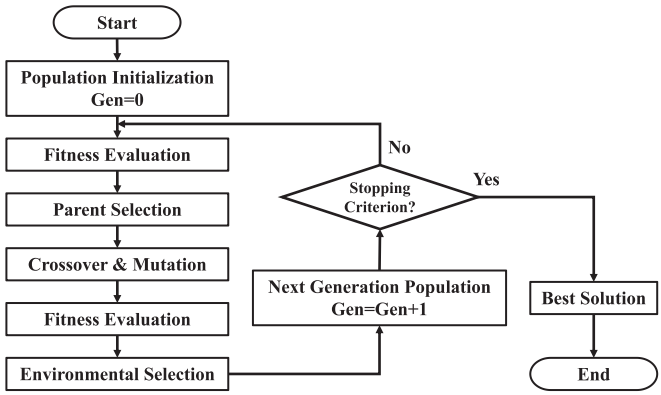

Fig. 1. Flow chart of genetic algorithm.

the secondary operator, reproduce offspring over several generations. In this work, we use ingenious ideas to overcome the issues of NAS on FER when using a variable-length encoding strategy, as adopted in this work, as opposed to the restrictive fixed-length representation used on FER tasks, as briefly discussed before. As shown in Fig. 1, the proposed algorithm follows the standard procedures of GA (Population initialization, fitness evaluation, crossover & mutation, and environmental selection). Specifically, we first form multiple individual network architectures using the proposed variable-length encoding strategy to form the population initialization. Next, we perform crossover and mutation operations on the populations to guide the population update. We then perform fitness evaluations on the individuals. Finally, the next generation population is derived by environmental selection.

III. Related Work

With the emergence of many challenging datasets such as RAF-DB and AffectNet in recent years, many researchers have begun to explore more diverse approaches for FER tasks.

Manually designed neural network architecture for FER: In the early stages of FER, Aouayeb et al. [42], proposed a FER system based on local binary patterns (LBP) as a feature extractor. Moreover, in this work, a local linear embedding technique is used to reduce the feature dimensionality, and a support vector machine is used for the classification part. Shan et al. [5] proposed a two-stage sparse learning framework. This is used to locate some common and specific information among different facial expressions, and then it performs the FER task. Recently, with the advancement of DL architectures, along with the proliferation of data generated through social media, more challenging and rich datasets are publicly available. Meng et al. [3], proposed Deep-Emotion, a DL approach based on attentional CNNs. The proposed framework focuses on the important parts of the face and achieves significant improvements over previous models. Meng et al. [10] proposed Emotion-FAN, which consists of the feature embedding module and the frame attention module for embedding face images into feature vectors and learning multiple attention weights for adaptive aggregation of feature vectors, respectively, to form a single discriminative video representation. Ding et al. [43] proposed FaceNet2ExpNet. This uses a novel distribution function to model the high-level neurons of the expression network from the model itself. In FaceNet2ExpNet a two-stage training algorithm is elaborated. In the pre-training phase, the convolutional layers of the expression network are trained and regularized by the face net, while, in the refinement phase, FaceNet2ExpNet adds fully connected layers to the pre-trained convolutional layers and trains the network jointly. Pourmirzaei et al. [44] proposed HMTL. This is similar to FaceNet2ExpNet in that it combines self-supervision as an auxiliary task merged into the supervised learning training environment. As an auxiliary task to supervised learning training, more information than labels can be obtained from self-supervised learning input data. With the emergence of large-scale in-the-wild datasets, many more effective methods have been proposed by researchers. For example, Wang et al. [11] proposed the Region Attention Network (RAN) to adaptively capture the importance of facial regions to occlusion and change thereby addressing the real-world problems of pose and occlusion robustness. However, RAN ignores the uncertainty in the dataset, and for this reason, Wang et al. [12] propose the Self-Cure Network (SCN) based on RAN, which effectively suppresses uncertainty and prevents the deep network from over-testing uncertain facial images. These above-mentioned methods, although performing well on the specified datasets, require a priori knowledge to design the network architecture with good performance. Thus, the ability to automatically design network architectures for FER tasks has become a pressing matter.

Automatically designed neural network architecture for FER: NAS was born with the growing need for automatically designed network architectures. Recently, some researchers have begun to explore the use of NAS in FER, achieving interesting results. For example, Aghera et al. [45], proposed MnasNet-FER, an automatic mobile neural architecture-based approach for FER tasks, which aims to search for a lightweight model using RL. However, as pointed out by the authors, this approach is costly in search and difficult to balance between obtaining a lightweight architecture while at the same time obtaining a well-performing network. Liu et al. [24] proposed DARTS, which transforms the space of discrete neural architectures into continuous differentiable forms and uses gradient optimization techniques to search for neural architectures. Although the time required is alleviated, this method requires a predetermined number of layers of the network architecture. Li et al. [46] proposed AutoFERNet based on DARTS, drawing on a simple but effective facial expression similarity (FES)-based rescaling method for in-the-wild datasets to alleviate the uncertainty problem caused by natural factors and annotator subjectivity. However, this approach is still based on DARTS and still suffers from the fact that it requires a priori knowledge and cannot be fully automated. To achieve full automation of architecture search, Evans et al. [30] proposed ConvGP based on genetic programming. ConvGP can exploit the flexibility of the GP representation. A program is automatically evolved to learn the coefficients of a convolutional filter and detect useful regions in an image, from which features are extracted to build a classifier. Unlike ConvGP, Masanori et al. [30] propose

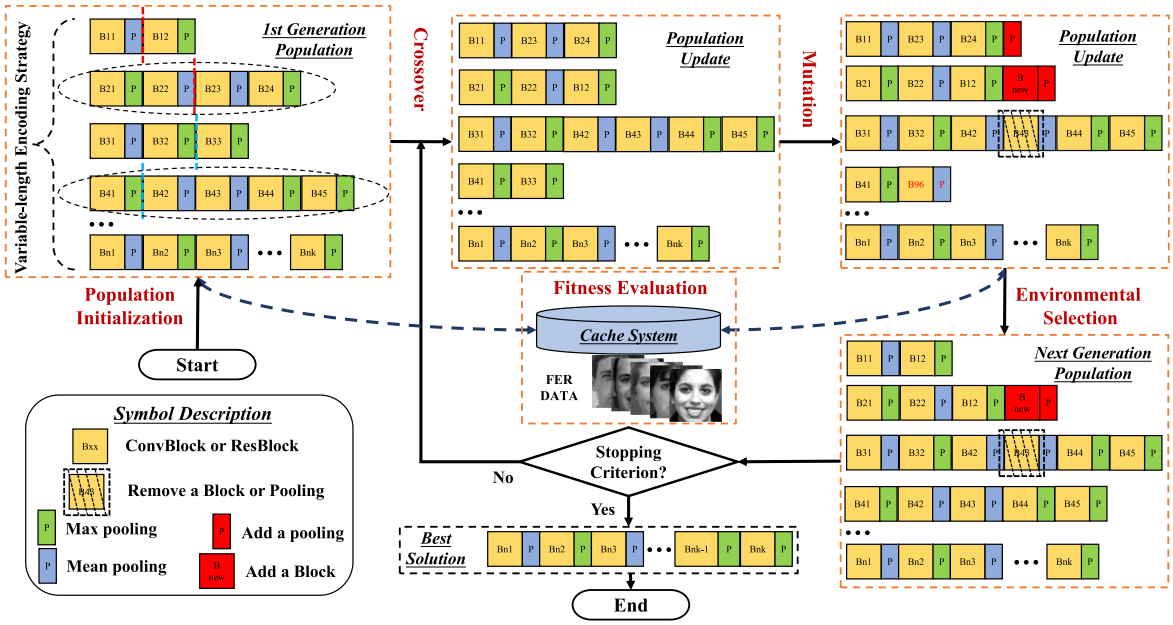

Fig. 2. General flowchart of ENAS-FERNet, whose main body follows the basic steps of the GA. Specifically, ENAS-FERNe encodes the neural network architecture as an individual in a population using the proposed variable-length encoding strategy, then performs the proposed crossover and mutation operations on the population, and finally selects the next generation population by environmental selection. Furthermore, we have designed two basic modules, ConvBlock and ResBlock, to encode the individuals in the population, a Cache System to improve the search efficiency involving fitness evaluation, and four mutation operations to find better individuals.

an automated CNN architecture for image classification tasks based on Cartesian Genetic Programming (CGP), which uses highly functional modules and the structure and connectivity of the CNN represented by the CGP encoding method is optimized to maximize verification accuracy. Although these methods of ConvGP and CGP successfully address the problem of DARTS requiring a priori knowledge, both suffer from the excessive time required.

IV. Evolutionary Neural Architecture Search for FER (ENAS-FERNet)

This section provides a detailed description of the proposed ENAS-FERNet, which can be divided into ENAS-FERNet (ConvBlock) and ENAS-FERNet (ResBlock) models. We first give a general introduction to the ENAS-FERNet, followed by the proposed variable-length encoding strategy. Then, population initialization and fitness evaluation are described, followed by the crossover and mutation operations. Finally, we describe the environmental selection of the populations.

A. Overview

Considering the respective characteristics of NAS and FER tasks, we design a generic automatic FER Network based on Evolutionary Neural Architecture Search (ENAS-FERNet). ENAS-FERNet follows the basic steps of the GA, and its general flow chart is shown in Fig. 2. Specifically, we first use a variable-length encoding strategy to encode the network architecture and thus complete the population initialization. Next, the proposed crossover and mutation operations are performed in the population. We choose the single-point crossover because it is a traditional crossover method with good performance [35]. Then, we perform the mutation operation on the network architectures represented by the individuals. Finally, the environmental selection is carried out on the parent population and the offspring population resulting from the crossover and mutation operations to form the next generation of populations.

In general, after initializing the population, the offspring individuals are generated by the NAS strategy (i.e., crossover and mutation operators). These offspring individuals are then trained to obtain their performance. Finally, environmental selection is used to produce the next generation population after all offspring have been trained. Therefore, the NAS strategy is applied before training the individual networks. It is worth noting that we have designed a cache system for evaluating the fitness of individuals in the population to improve the efficiency of the search. If the network architecture represented by an individual in the population is already present in the cache system, then the fitness of the individual can be obtained without further computation, while in the opposite case, the fitness of the individual needs to be computed in conjunction with the dataset. The implementation of the crossover, mutation and environmental selection operations involved in ENAS-FERNet will be visualized in Section IV-C.

B. Variable-Length Encoding Strategy

The fixed-length encoding strategy was often employed for GAs in the early NAS works, which was mainly influenced by the fixed length of individual chromosomes in nature. For example, GeNet used a fixed-length string to represent the network architecture and still suffered from the shortcomings

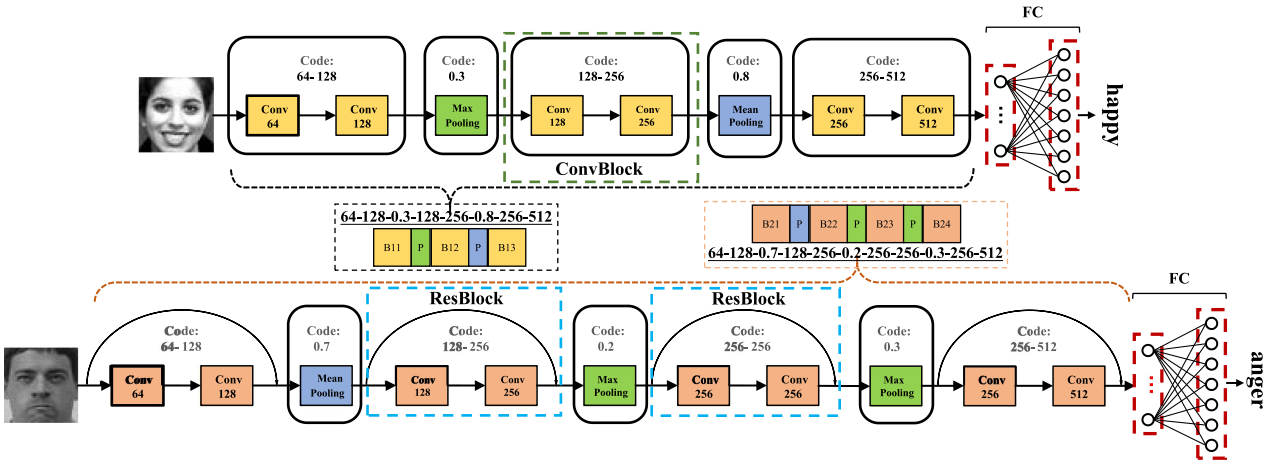

Fig. 3. Top: A neural network architecture is formed by stacking 3 ConvBlocks and 2 pooling layers, where a ConvBlock represents two convolutional layers and we use a number between $(0, 0.5)$ for the max pooling layer and a number between $[0.5, 1)$ for the mean pooling layer. Thus, the string "64-128-0.3-128-256-0.8-256-512" represents a neural network with a depth of 8. Bottom: A neural network architecture is formed by stacking 4 ResBlocks and 3 pooling layers, where a ResBlock represents two convolutional layers combined with a skip connection. Therefore, the string "64-128-0.7-128-256-0.2-256-256-0.3-256-512" represents a neural network with a depth of 11. Among them, "FC" represents the fully connected layer.

of the fixed encoding strategy. In this case, the depth of the encoded network architecture had to be predefined in advance. However, in the ideal case, we do not know the depth of the optimal network architecture, and the fixed encoding strategy leads to the problem that the final obtained network architecture may be incorrectly estimated. Although many researchers have designed many variable-length encoding strategies, the resulting network architecture is still not optimal and not matched because the authors did not redesign the crossover operation accordingly or suffer from achieving precision [35]. For example, binary encoding is the widely used variable-length encoding strategy, but it requires effort between binary conversions and suffers from Hamming cliffs resulting in reduced accuracy [47]. In the value encoding strategy, the chromosome can be represented using a specific string such as a character and integer number [48]. Although the value encoding strategy is mainly used in solving problems with more complicated values, it still lacks the crossover operations corresponding to its encoding strategy [48]. For the above reasons, we propose a variable-length encoding strategy and the corresponding crossover operator to improve the performance of the algorithms and solve their inefficiencies in terms of accuracy.

Each individual represents a valid neural architecture for the FER tasks. The variable-length encoding strategy is used to construct an individual. This encoding strategy forms the network architecture by stacking the basic unit blocks in the designed search space. Specifically, the search space consists of the following basic units: 3×3 convolution, 2×2 max pooling, 2×2 mean pooling, and the skip connection. With these basic units, we constructed two basic modules, ConvBlock and ResBlock, where a ConvBlock consists of two basic principal convolutional layers. It is worth noting that we have combined the skip connection with the ConvBlock, which in turn forms the ResBlock. The skip connection is able to connect the neurons of the layers that are not adjacent. The skip connection was first proposed by Gers et al. [49] and often used as a gate

TABLE I
THE SEARCH SPACE IS FORMED BY STACKING THE BASIC UNITS CONVBLCOK OR RESBLCOK, MAX POOLING AND MEAN POOLING. IN THE TABLE, C STANDS FOR NUMBER OF OUTPUT CHANNELS, K STANDS FOR RECEPTIVE FIELD SIZE (KERNEL SIZE), AND S STANDS FOR STRIDE SIZE

Unit type	Variation
ConvBlcok (C,K,S)	$C \in \{64, 128, 256, 512\}$ $K \in \{3 \times 3\}$ $S \in \{1 \times 1\}$
ResBlcok (C,K,S)	$C \in \{64, 128, 256, 512\}$ $K \in \{3 \times 3\}$ $S \in \{1 \times 1\}$
max pooling (K,S)	$K \in \{2 \times 2\}$ $S \in \{2 \times 2\}$
mean pooling (K,S)	$K \in \{2 \times 2\}$ $S \in \{2 \times 2\}$

mechanism, which can effectively train neural networks such as long short-term memory [50] and avoid the vanishing gradient problem [49]. The ConvBlock or ResBlock is stacked with pooling layers to form a neural network architecture, as shown in Fig. 3. The number of feature maps of the CNN architecture layers can be varied so that they can be set to 64, 128, 256, 512, and the stride size is defined to 1×1, inspired by the ResNet series [51]. The pooling layer is divided into mean pooling and max pooling, and the step size is set to 2×2. These basic units can be presented more clearly in Table I. The best potential neural network architecture can be searched without knowing the optimal structure of the network, which is achieved by our proposed variable-length encoding strategy.

A population consists of many individuals formed by the variable-length encoding strategy. The individual shown in Fig. 3 represents an example of a CNN architecture using the variable-length encoding strategy, where the top of Fig. 3 represents a network architecture formed by stacking ConvBlocks and pooling layers, and the bottom of Fig. 3 represents a network architecture formed by stacking ResBlocks and pooling layers. It should be noted that the code above the ConvBlock or ResBlock

represents the number of corresponding feature maps in the relevant convolutional layer, and the code above the pooling layer represents the max pooling layer if the number is between (0,0.5) and the mean pooling layer if it is between [0.5,1). Thus, we can see that the string "64-128-0.3-128-256-0.8-256-512" at the top of Fig. 3 represents a network architecture with a depth of 8 formed by 3 ConvBlocks and 2 pooling layers; the string "64-128-0.7-128-256-0.2-256-256-0.3-256-512" at the bottom of Fig. 3 represents a network architecture with a depth of 11 formed by 4 ResBlocks and three pooling layers. By evolving this, our automatic facial expression recognition network is able to explore more diverse network architectures, including networks of different depths. This is an important element since deeper networks have been reported to generate better results on multiple tasks so that our proposed approach handles this naturally. This is a crucial element differentiating our approach to other methods that use fixed-length representations that do not allow for the depth of a network to adapt during evolution.

C. Population Initialization

The initialized search space is a subspace of the search space i.e., it determines which types of network architectures can appear in the initialized population [32]. There are three main approaches to determine the initialization search space in the search space, namely, trivial space [28], random space [33], and well-designed space [52]. The first two can be considered as directly designed initialization population operations, and although in principle they can contain many network architectures, the search space is often exponential and time-consuming to search. For this reason, our method can be considered as the latter for initializing populations and we constructed two types of initialized network architecture models, ENAS-FERNet (ConvBlock) and ENAS-FERNet (ResBlock) as shown in Fig. 3. We use ConvBlock or ResBlock and pooling layer stacking to form the network architecture, which is then represented as individuals in the population. The proposed variable-length encoding strategy is then used to continuously encode the network architecture with different lengths until the number of populations is reached as shown in Fig. 2, which leads to the population initialization.

D. Fitness Evaluation

After the population initialization is completed, we train and validate the network architecture represented by the individuals in the population on the specified dataset, and use the classification accuracy as the fitness value of corresponding individual. The network architecture and the corresponding fitness value are then stored in the designed cache system. If the network architecture of an individual is validated in subsequent populations, we first request the cache system before training the network architecture, and if the same network architecture exists in the cache system, then the corresponding fitness value can be obtained directly without extra training. This avoids redundant calculations and thus improves the search efficiency.

E. Crossover and Mutation

After the population initialization and fitness evaluation, we update the population using the proposed algorithm. The details of crossover, mutation, and environmental selection are shown in Algorithm 1. Specifically, the first part corresponds to crossover (Lines 1-18), immediately followed by mutation (Lines 19-26) and finally succeeded by environmental selection (Lines 27-37). In this subsection, we focus on the description of crossover and mutation operations in population updating, and the environmental selection is presented in the next subsection. During the crossover operation, there will be $|P_t|$ offspring produced. Specifically, two parents are first selected (Lines 3-7). This binary tournament selection and it is popular in GAs for single-objective optimization. Once the parent individuals are selected, we generate a random number between 0 and 1 and determine whether to perform the crossover operation based on a predefined probability. If the generated random number is less than the predefined probability, the network architecture represented by the parent individual is divided into two parts, and the two parents exchange the corresponding parts with each other to form offspring individuals (Lines 9-14). If this is not the case, the selected individuals are directly put into Q_t as offspring individuals (Line 16).

When performing mutation operations, a random number r is generated first (Line 20), which plays the role of judging whether or not to proceed to perform mutations. Specifically, if r is below than p_m (Lines 21-25), the mutation operation will be performed on the current individual. When the conditions are met to mutate an individual, a position i is randomly selected from the current individual. At the same time one mutation operation *op* is selected from the defined mutation list l_m, which is based on the probabilities predefined by p_l. Then, the mutation operator *op* is performed on the selected position *i*. The mutation operators are defined in l_m as follows.

i) Adding a random pooling layer to the network architecture at the selected position.
ii) Adding a random Block with a random initial value to the network architecture at the selected position.
iii) Removing pooling or Block layer at the selected position.
iv) Changing the pooling layer to another type or changing the value of Block layer at the selected position.

It is worth noting that the optimal depth can be found through mutation operators. Particularly, we denote the first two mutation operators as the "depth-increasing operator", which provides a possibility to increase the depth of the network architecture represented by an individual, while the third mutation operator provides a possibility to decrease the depth, which is denoted as the "depth-decreasing operator". For example, if *N* is the optimal depth of network architecture represented by an individual, after the random initialization, some individuals may have the depth of network architecture smaller than *N*, while others are greater than *N*. During evolution, each individual has a chance to be mutated. If the depth of the individual is greater than *N* and the depth-decreasing operator is selected, the depth of individual will be decreased towards *N*, and *vice versa*. In general, when the population is randomly initialized, the depth of the optimal

Algorithm 1: Population Updating.

Input: The population P_t containing individuals with fitness, the probability for crossover operation p_c, the probability for mutation operation p_m, the mutation operation list l_m, the probabilities of selecting different mutation operations p_l
Output: The population for next generation P_{t+1}

```
// Crossover
Q_t ← φ
while |Q_t| < |P_t| do
    M_1 ← Randomly select two individuals from P_t and then select the one with better fitness
    M_2 ← Repeat Line 3
    while M_2 == M_1 do
        Repeat Line 4
    end
    r ← Randomly generate a number from (0,1)
    if r < p_c then
        Divide M_1 into two random parts by a point
        Divide M_2 into two random parts by a point
        newM_1 ← 1st part of M_1 adds 2nd part of M_2
        newM_2 ← 1st part of M_2 adds 2nd part of M_1
        Q_t ← Q_t ∪ newM_1 ∪ newM_2
    else
        Q_t ← Q_t ∪ M_1 ∪ M_2
    end
end
// Mutation
foreach individual M do
    r ← Randomly generate a number from (0,1)
    if r < p_m then
        i ← Randomly choose a point in M
        op ← Select one operation from l_m based on the probabilities in p_l
        Mutate op at the point i of M
    end
end
// Environmental Selection
P_{t+1} ← φ
while |P_{t+1}| < |P_t| do
    M_1, M_2 ← Two random individuals from Q_t ∪ P_t
    M_better ← The better one from M_1 ∪ M_2
    P_{t+1} ← P_{t+1} ∪ M_better
end
M_best ← The best one from Q_t ∪ P_t
if M_best is not in P_{t+1} then
    Replace the worst one in P_{t+1} by M_best
end
Return P_{t+1}
```

individual network can be found by performing either the depth-increasing operator or the depth-decreasing operator.

F. Environmental Selection

Finally, we use a binary tournament selection and elitist strategy for environmental selection to form the next generation population. The former method randomly selects two individuals among the parent population and the offspring and then selects the better individual as the parent of the next generation until the number of selected individuals is equal to the population size (Lines 28-32). The latter operator works as follows: if the best individual from the previous generation is not in the next generation, then we replace the worst individual in the generation with the best individual to form the new generation of parents (Lines 33-36). In principle, an ideal population should contain not only good individuals but also relatively poor individuals to enhance population diversity. For this purpose, binary tournament selection is often utilized. However, using only binary tournament selection may miss the best potential individual, resulting in an algorithm that fails to evolve toward a better direction. Therefore, we explicitly employ elitism, as explained before.

V. Experiments and Discussion of Results

In this section, we first describe the datasets used in our work and our implementation details in the first two subsections. We then validate the effectiveness of the proposed ENAS-FERNet on representative laboratory-controlled and in-the-wild datasets. Finally, we present ablation studies to better understand ENAS-FERNet.

A. Datasets

To evaluate our method, we use two representative datasets under laboratory-controlled conditions, namely CK+ [53] and JAFFE [54], [55], and two popular in-the-wild facial expression datasets, namely RAF-DB [6] and AffectNet [7]. These datasets cover different scales of face images, challenging conditions, balanced and unbalanced gender images. Thus, these datasets represent problems of various degrees of difficulty, size, and dimensionality reasonably well. Moreover, we carefully choose these benchmark problems so that our extensive evaluations of the proposed methods (along with those used for comparison purposes) are not problem dependent allowing us to draw a reasonable conclusion.

CK:. The extended Cohn-Kanade (CK+) dataset [53] includes a total of 593 video sequences across 123 different subjects, which ranges in age from 18 to 50 years old, of various genders and backgrounds. Each video shows a facial transition from neutral to other expressions, where the recording speed is 30 frames per second (FPS) and the final result is a resolution of 640x490 or 640x480 pixels per image. In these videos, there are a total of 327 images labeled as one of seven expression categories: anger, contempt, disgust, fear, happiness, sadness, and surprise. It is worth noting that the CK+ dataset is currently the most extensively used laboratory-controlled dataset in the world and is also the experimental dataset chosen for the majority of facial expression recognition methods. We divided the dataset into the training set, validation set, and testing set, as 687, 101, and 193 images, respectively.

JAFFE: The JAFFE dataset [54], [55] comprises a total of 213 images, which cover different facial expressions from ten different Japanese female subjects. Each subject was required to make seven facial expressions and 60 annotators annotated the

images with average semantic scores for each facial expression. This dataset is currently applied to most facial expression recognition methods as a benchmark dataset. In our experiments, we used 120 images for training, 23 images for validation, and 70 images for testing (10 images per emotion in the test set, plus 10 images for the neutral expression).

RAF-DB: The Real-world Affective Faces Database (RAF-DB) [6] is a typical dataset for in-the-wild facial expression, which consists of 29672 facial images labeled with basic or compound expressions by 40 independent annotators. This dataset contains a wide range of rich scenarios that closely resemble realistic facial expressions, e.g. it covers people of different ages, genders and ethnicities, and has gestural variations, slight perturbations, different coupling situations (e.g. glasses, facial hair or self-occlusion) and post-processing operations (e.g. various filters and special effects) in the expressions. It is currently the most widely used in-the-wild dataset and is the experimental dataset chosen by most methods. In our experiment, only images with basic emotions were used, including 12,271 images as training data and 3,068 images as testing data.

AffectNet: The AffectNet [7] is a large in-the-wild dataset of approximately one million images, obtained by searching the internet using keywords. There are around 400,000 images in AffectNet that have been manually labeled for the presence of eight facial expressions (neutral, happy, angry, sad, fear, surprise, disgust, and contempt) and the intensity of their value and evocation. Specifically, AffectNet has an imbalanced testing set, a balanced validation set, and an imbalanced training set. We report mean class accuracy on the validation set where each category contains 500 samples.

B. Implementation Details

The GA used by ENAS-FERNet followed the commonly used settings [35], with population size and the generations to 20, and the crossover and mutation probabilities to 0.9 and 0.2, respectively. Recall that we use different types of mutations, we define 0.9 for adding a block (ConvBlock or ResBlock), and 0.1 for the rest of the three mutations, as described in Section IV. When the verification accuracy is as expected or the evolutionary process reaches the max number of generations, we terminate the algorithm. For network training, we resize all training data to 48×48, set the batch size to 64, and choose stochastic gradient descent (SGD) as the optimization method. The momentum of SGD is set to 0.9 and the weight decay of SGD is set to 3e-4. For laboratory-controlled datasets such as CK+ and JAFFE, we trained a total of 600 epochs. We set the initial learning rate to 0.025 and adjusted the learning rate to 0.017 at the 100th epoch, 0.001 at the 300th epoch, and 0.0001 at the 500 epoch. For in-the-wild datasets such as RAF-DB and AffectNet, we trained a total of 100 epochs and set the initial learning rate to 0.1, and divide it by 10 after 60 epochs and 80 epochs. These detailed hyperparameter settings can be presented more clearly in Table II. All experiments were performed on an Ubuntu server with an NVIDIA 2080 Ti GPU card using Pytorch. The results reported and discussed in the following section, are the result of five runs for each of the datasets described before.

TABLE II
THE SETTINGS OF THE RELEVANT HYPERPARAMETERS USED IN THE EXPERIMENT

Hyperparameters	Value
Crossover probability	0.9
Mutation probability	0.2
Population generation	20
Population size	20
Optimizer	SGD
The momentum of SGD	0.9
The weight decay of SGD	3e-4
Initial learning rate	0.025 / 0.1
The number of epochs	600 / 100

C. Comparison With Peer Competitors

In this subsection, we evaluated the proposed ENAS-FERNet on both laboratory-controlled and in-the-wild datasets, comparing its performance with state-of-the-art (SOTA) methods that were trained from scratch.

Comparison with peer competitors on laboratory-controlled facial expression datasets: To show the effectiveness and efficiency of the proposed algorithm, we selected the SOTA algorithms as peer competitors, which are trained from scratch. Among these methods, manually designed network architectures are Deep-Emotion [3], Ensemble Multi-feature [56], LBP [42], SAFER [57], Emotion-FAN [10], FFER [58], HMTL [44], and Vision Transformer-based methods [59]. Automatically designed neural network architectures are gradient-based automated NAS methods such as Auto-FERNet [46] and DARTS [24] and genetic programming-based automated NAS methods such as ConvGP [30] and CGP [30]. These methods include both manually designed neural network architectures and automatically designed neural network architectures, representing the best neural network architectures for the field of facial expression recognition, trained from scratch.

As shown in Table III, through experimental comparative analysis, our approach was trained from scratch and did not use additional data. Our ENAS-FERNet (ResBlock) approach achieved 100% accuracy on the CK+ dataset, reaching the best-known results to date, and 95.71% accuracy on the JAFFE dataset, higher than most methods. For the CK+ dataset, most methods can achieve around 99% accuracy due to its small size and the high quality of the dataset, and our proposed method is able to break the 1% bottleneck due to the ability of our method to automatically evolve a better network architecture for a specific dataset. Our ENAS-FERNet (ResBlock) is only 1.43% behind the current SOTA approach on the JAFFE dataset and is higher than the majority of methods. This may be due to the fact that the JAFFE dataset is too small for a particular image to be recognized, that other methods have special settings for this dataset, and that our general-purpose method has a slightly lower accuracy classification rate. Notably, our method ENAS-FERNet (ConBlock), which is based on ConBlock as the basic block, achieves 99.49% and 97.14% classification accuracy on the CK+ and JAFFE datasets in only 13.33 and 11.00 hours, respectively. Auto-FERNet achieved 98.89% classification accuracy in 19.00 hours on the CK+ dataset and its time and classification accuracy were lower than those of ENAS-FERNet

TABLE III
COMPARISON WITH PEER COMPETITORS IN THE CASE OF WITHOUT TRAINING FROM SCRATCH ON LABORATORY-CONTROLLED FACIAL EXPRESSION DATASETS (CK+ AND JAFFE) THE '+' AND '-' SYMBOLS OF CONVGP ARE USED TO INDICATE THAT THE AVERAGE TESTING ACCURACY OF THE PROPOSED METHOD IS, RESPECTIVELY, SIGNIFICANTLY BETTER AND SIGNIFICANTLY WORSE COMPARED TO THAT OF A GIVEN CLASSIFIER

Method	CK+ (%)	JAFFE (%)	Search Cost (h)	Manual assistance?
Deep-Emotion	98.00	92.80	Manual	Completely needed
Ensemble Multi-feature	-	80	Manual	Completely needed
LBP	93.9	88.3	Manual	Completely needed
SAFER	96.37	95.30	Manual	Completely needed
Emotion-FAN	99.69	-	Manual	Completely needed
AFER	-	96.05	Manual	Completely needed
HMTL	98.23	79.88	Manual	Completely needed
ViT	98.17	94.83	Manual	Completely needed
ViT + SE	99.80	92.92	Manual	Completely needed
ViT*	98.57	88.23	Manual	Completely needed
ViT + SE*	99.49	90.61	Manual	Completely needed
Auto-FERNet	98.89	97.14	19.00 / 19.00	Partially needed
DARTS	99.85	92.86	19.32 / 3.46	Partially needed
ConvGP	-	96.67	- / 18.18	Completely not needed
ConvGP (No pooling)	-	90.00	- / 43.61	Completely not needed
CGP	98.45	90.00	12.00 / 19.25	Completely not needed
ENAS-FERNet (ConvBlock)	99.49	97.14	13.33 / 11.00	Completely not needed
ENAS-FERNet (ResBlock)	100.00	95.71	29.04 / 27.84	Completely not needed

(ConvBlock). Auto-FERNet achieved 97.14% classification accuracy in 19.00 hours on the JAFFE dataset. Although the classification accuracy of ENAS-FERNet (ConvBlock) was the same as Auto-FERNet, our time cost was only about 1/2 of Auto-FERNet. Furthermore, DARTS achieved 99.85% classification accuracy in 19.32 hours on the CK+ dataset. Although the classification accuracy of ENAS-FERNet (ConvBlock) was 0.36% lower than DARTS, ENAS-FERNet (ConvBlock) took about 11 hours less than DARTS. DARTS achieves 92.86% classification accuracy in 3.46 hours on the JAFFE dataset, which is lower than the proposed ENAS-FERNet (ConvBlock) despite its inherent advantage in speed based on gradient. From the perspective of being completely free of manual assistance, the ConvGP method achieves the highest classification accuracy of 96.67% in 18.18 hours on the JAFFE dataset. It is lower than the 97.14% achieved by ENAS-FERNet (ConvBlock), and the time required by ENAS-FERNet (ConvBlock) is only about 1/2 of ConvGP. ConvGP (No pooling) achieves 90.00% in 43.61 hours on the JAFFE dataset. ENAS-FERNet (ConvBlock) is 7.14% more accurate than ConvGP (No pooling) and takes only 1/4 of ConvGP (No pooling)'s time. The classical CGP method achieves 98.45% accuracy in 12.00 hours on the CK+ dataset and 90.00% in 19.25 hours on the JAFFE dataset, which are lower than the 99.49% and 97.14% accuracy achieved by the proposed ENAS-FERNet (ConvBlock). In addition, ENAS-FERNet (ConvBlock) takes about the same time as the CGP on the CK+ dataset but is only about 1/2 of the CGP on the JAFFE dataset.

In general, ENAS-FERNet (ConvBlock) and ENAS-FERNet (ResBlock) can achieve competitive results on both CK+ and JAFFE datasets, which indicates that they have good generalization capability, and ENAS-FERNet (ResBlock) can achieve better results compared with ENAS-FERNet (ConvBlock) on the CK+ dataset, which is due to the fact that ResBlock can greatly eliminate the difficulty of training neural networks with too much depth and find a deeper neural network architecture. Meanwhile, the manually designed neural network architectures require manual assistance completely, and the automatically designed neural architectures require partial manual assistance such as Auto-FERNet and DARTS, which still rely more or less on manual assistance. Although such methods as ConvGP and CGP can also be completely free of manual assistance, these methods are less effective and time-consuming. Our method is able to obtain good performance completely without manual assistance. Overall, existing methods have poor generalization capability. On the contrary, our proposed algorithm performs well on all datasets, which also indicates better generalizability of ENAS-FERNet.

Comparison with peer competitors on in-the-wild facial expression datasets: While the above experiments demonstrate that our approach ENAS-FERNet has achieved good results on laboratory-controlled datasets. Realistic facial expression recognition is more challenging and relevant to our daily lives, so it is essential to validate our method on in-the-wild datasets. Here we focus on two of the most popular in-the-wild datasets and compare our approach ENAS-FERNet with their peer competitors. We also selected the SOTA algorithms as peer competitors, which are trained from scratch. Among these methods, manually designed network architectures are CurriculumNet [60], MetaCleaner [61], ResNet18 [51], SCN [12], VGG-16 [62], and EfficientFace series [63]. Automatically designed neural network architectures are gradient-based automated methods such as DARTS and genetic programming-based automated search methods such as CGP. These methods represent the best networks in manual and automated neural network architectures in the case of training from scratch.

Compared with the manually designed neural network architecture as shown in Table IV, ENAS-FERNet (ConvBlock) achieved 83.87% accuracy on the RAF-DB dataset, outperforming the existing manually designed neural network approach and establishing a new SOTA result in the case of training from scratch. The accuracy of FERNet (ResBlock) is 83.41%, which is higher than most of the manually designed methods, and achieves competitive results. On the AffectNet dataset,

TABLE IV
THE PROPOSED ENAS-FERNET COMPARES WITH PEER COMPETITORS IN THE CASE OF TRAINING FROM SCRATCH ON RAF-DB AND AFFECTNET DATASETS WHILE THE LFE STANDS FOR LOCAL-FEATURE EXTRACTOR AND CSM STANDS FOR CHANNEL-SPATIAL MODULATOR, RESPECTIVELY

Method	Train from scratch	RAF-DB (%)	AffectNet (%)	Search Cost (h)	Manual assistance?
CurriculumNet	✓	74.67	-	Manual	Completely needed
MetaCleaner	✓	77.18	-	Manual	Completely needed
ResNet18	✓	72.00	46.58	Manual	Completely needed
SCN	✓	78.31	47.28	Manual	Completely needed
VGG-16	✓	80.96	51.11	Manual	Completely needed
EfficientFace (Baseline)	✓	82.23	-	Manual	Completely needed
EfficientFace (LFE)	✓	83.57	-	Manual	Completely needed
EfficientFace (CSM)	✓	83.12	-	Manual	Completely needed
EfficientFace (LFE+CSM)	✓	83.83	-	Manual	Completely needed
DARTS	✓	79.27	48.80	28 / 106	Partially needed
CGP	✓	76.73	47.05	124 / 619	Completely not needed
ENAS-FERNet (ConvBlock)	✓	83.87	53.15	36 / 149	Completely not needed
ENAS-FERNet (ResBlock)	✓	83.41	50.05	49 / 170	Completely not needed

ENAS-FERNet (ConvBlock) with accuracy of 53.15% and ENAS-FERNet (ResBlock) with accuracy of 50.05% achieved competitive results among the manually designed neural network architectures. Compared to automated methods, for example, DARTS achieved 79.27% accuracy in 28 hours on the RAF-DB dataset. Although DARTS is faster due to its gradient approach, DARTS requires partial manual assistance and cannot be fully automated. 79.27% accuracy of DARTS is lower than ENAS-FERNet (ConvBlock) with 83.87% and ENAS-FERNet (ResBlock) with 83.41%. DARTS achieved 48.80% accuracy in 106 hours on AffectNet dataset, which is lower than ENAS-FERNet (ConvBlock). CGP achieved the accuracy of 76.73% on the RAF-DB dataset in 124 hours. Although CGP can be fully automated without manual assistance, the accuracy of CGP is lower than ENAS-FERNet (ResBlock). Specifically, CGP with the accuracy of 76.73% is lower than the accuracy of 83.87% for ENAS-FERNet (ConvBlock) and 83.41% for ENAS-FERNet (ResBlock). And the CGP accuracy of 124 hours is much higher than that 36 hours for ENAS-FERNet (ConvBlock) and 49 hours for ENAS-FERNet (ResBlock). Furthermore, DARTS achieved an accuracy of 48.80% on the AffectNet dataset in 106 hours. Although DARTS is faster due to the gradient method, DARTS requires some manual assistance and cannot be fully automated, and the accuracy of 48.80% is lower than ENAS-FERNet (ConvBlock). CGP achieved 47.05% accuracy on the AffectNet dataset in 619 hours, although the CGP method can be fully automated, the accuracy of CGP 47.05% is lower than the accuracy 53.15% of ENAS-FERNet (ConvBlock) and 50.05% of ENAS-FERNet (ResBlock). It is worth noting that CGP takes 619 hours on AffectNets dataset, while our ENAS-FERNet takes at most 170 hours and yields better results, significantly reducing search costs and improving performance.

When we focus our attention on the RAF-DB and AffectNet datasets, we see a good number of manually designed methods that perform relatively well on these datasets. For example, CurriculumNet designs training curriculum by measuring data complexity using cluster density which can avoid training noisy-labeled data in the early stages. MetaCleaner aggregates the features of several samples in each class into a weighted mean feature for classification which can also weaken the noisy-labeled samples. SCN suppresses the uncertainties efficiently and prevents deep networks from overfitting uncertain facial images. Both the improved VGG-16 and ResNet18 have good results. EfficientFace presents an efficiently robust FER network, which holds much fewer parameters but is more accurate and robust to the FER in the wild. All manually designed networks: CurriculumNet, MetaCleaner, SCN, and EfficientFace improve the baseline largely but are still inferior compared to the ENAS-FERNet (ConvBlock) and ENAS-FERNet (ResBlock), which automatically finds a suitable architecture yielding competitive results. Overall, ENAS-FERNet achieves the best results among its peer competitors on the RAF-DB and AffectNet datasets. Specifically, ENAS-FERNet (ConvBlock) outperforms the best manual method by 0.04% and 2.04%, respectively. In addition, there is a significant improvement over classical networks such as ResNet18 and VGG-16.

In this subsection, we experimentally validate and analyze the proposed ENAS-FERNet on two representative laboratory-controlled datasets and two typical in-the-wild datasets, respectively. The comprehensive analysis and results show that the proposed ENAS-FERNet not only has the ability to search out the well-performing network architecture automatically and efficiently, but also has good generalization capability.

D. Ablation Studies

After the comparative analysis of the above experimental results, ENAS-FERNet performs better or is competitive to most SOTA approaches in the specialized literature. To better understand why ENAS-FERNet performs incredibly well and for a more comprehensive analysis, we focus on the ability of ENAS-FERNet to handle uncertainty (using noisy datasets as a measure of uncertainty) on the one hand, and on the other hand we compared the performance of ENAS-FERNet during the evolutionary process. Finally, we present a comprehensive comparative analysis of ENAS-FERNet.

The ability of ENAS-FERNet to handle uncertainty: We next perform some validation analysis on the representative dataset, RAF-DB, which is currently the most widely used in-the-wild dataset and is the experimental dataset chosen by most methods [12]. The in-the-wild datasets are more challenging such as uncertainty, and the current prevailing approaches are mainly based on the manually designed network architecture. The uncertainties of in-the-wild datasets mainly come from

TABLE V
THE PROPOSED METHOD ENAS-FERNET IS EXPERIMENTALLY COMPARED WITH SIMILAR COMPETITORS ON RAF-DB DATASETS WITH SYNTHETIC NOISE PERCENTAGE OF 10% AND 20%, WHICH ARE COMPARED IN THE CASE OF TRAINING FROM SCRATCH

Method	Train from scratch	RAF-DB (10%)	RAF-DB (20%)	Search Cost (h)	Manual assistance?
CurriculumNet	✓	68.5	61.23	Manual	Completely needed
MetaCleaner	✓	68.45	61.35	Manual	Completely needed
ResNet18	✓	61.43	55.50	Manual	Completely needed
ResNet18+SCN	✓	70.26	63.50	Manual	Completely needed
DARTS	✓	76.11	75.95	29 / 28	Partially needed
CGP	✓	75.16	68.61	125 / 124	Completely not needed
ENAS-FERNet (ConvBlock)	✓	76.63	75.20	36 / 29	Completely not needed
ENAS-FERNet (ResBlock)	✓	77.05	74.84	62 / 72	Completely not needed

TABLE VI
A COMPREHENSIVE COMPARISON ANALYSIS TABLE OF ENAS-FERNET WITH MANUAL AND AUTOMATED METHODS, IN WHICH MANUAL * REPRESENTS THE BEST RESULT OF MANUALLY DESIGNED NEURAL NETWORK ARCHITECTURE AND AUTOMATIC* REPRESENTS THE BEST RESULT OF AUTOMATICALLY DESIGNED NEURAL NETWORK ARCHITECTURE. ± MANUAL* OR ±AUTOMATIC* RESPECTIVELY REPRESENTS HOW MUCH HIGHER OR LOWER THE PROPOSED ENAS-FERNET IS THAN THE BEST RESULT OF THE MANUALLY OR AUTOMATICALLY DESIGNED NEURAL NETWORK ARCHITECTURE

Dataset (Noise)	CK+	JAFFE	RAF-DB	AffectNet	RAF-DB (10%)	RAF-DB (20%)
DARTS (%, h)	99.85 / 19.32	92.86 / 3.46	79.27 / 28	48.80 / 106	76.11 / 29	75.95 / 28
CGP (%, h)	98.45 / 12.00	90.00 / 19.25	76.73 / 124	47.05 / 619	75.16 / 125	68.61 / 124
ENAS-FERNet (ConvBlock) (%, h)	99.49 / 13.33	97.14 / 11.00	83.87 / 36	53.15 / 149	76.63 / 36	75.20 / 29
ENAS-FERNet (ResBlock) (%, h)	100.00 / 29.04	95.71 / 27.84	83.41 / 49	50.05 / 170	77.05 / 62	74.84 / 72
Manual* (%)	99.80	96.05	83.83	51.11	70.26	63.50
Automatic* (%)	99.85	97.14	79.27	48.80	76.11	75.95
(±Manual*, ±Automatic*)	(+0.20, +0.15)	(+1.09, +0.00)	(+0.04, +4.60)	(+2.04, +4.35)	(+6.37, +0.94)	(+11.70, -0.75)

ambiguous facial expressions, low-quality facial images, inconsistent annotations, and incorrect annotations (i.e., noisy labels) [12]. Considering that only the labels can be specifically quantified and analyzed, we explored ENAS-FERNet at two levels of noise ratio of 10% and 20% to verify its robustness. Specifically, we randomly choose 10% and 20% of training data for each category and randomly change their labels to others. All peer competitors selected for comparison were trained from scratch, ensuring a fair comparison.

As shown in Table V, the accuracy of ENASFERNet (ConvBlock) is 76.63% and 75.20%, corresponding to 36 hours and 29 hours, respectively, at the RAF-DB levels of 10% and 20%. The accuracy of the proposed ENASFERNet (ResBlock) is 77.05% and 74.84%, corresponding to 62 hours and 72 hours, respectively, at the RAF-DB levels of 10% and 20%. Compared to the manually designed neural network architecture, ENASFERNet (ConvBlock) and ENASFERNet (ResvBlock) outperformed the second place by 6.37% and 6.79% at 10% noise ratio, 11.70% and 11.34% at 20% noise ratio. ENAS-FERNet (ConvBlock) and ENASFERNet (ResvBlock) do not require any manual assistance at all and are able to obtain good performance. Compared to the automatically designed neural network architectures, for example, DARTS requiring partial manual assistance obtained accuracies of 76.11% and 75.95% at RAF-DB levels of 10% and 20%, corresponding to 29 and 28 hours, respectively. ENASFERNet (ConvBlock) and ENASFERNet (ResBlock) have higher accuracy than DARTS by 0.52% and 0.94% at 10% noise ratio, and new SOTA result is established in this case of training from scratch. The design is fully automatic and does not require manual assistance. For example, the accuracy of CGP with no human help at RAF-DB level of 10% and 20% is 75.16% and 68.61%, corresponding to 125 hours and 124 hours, respectively. The accuracies of ENASFERNet (ResBlock) were higher than that of CGP by 1.89% and 6.32% at a noise ratio of 10% and 20%, and the time spent by ENASFERNet (ResBlock) were 62 hours and 36 hours, respectively, which were about 1/2 and 1/4 of the time spent by CGP.

In general, ENAS-FERNet (ConvBlock) and ENAS-FERNet (ResBlock) can achieve good results on the RAF-DB dataset with noise ratios of 10% and 20%, which indicates that they have good generalization ability and also indicates that the fused skip connection network can train a deeper network architecture and achieve better results. FERNet (ConvBlock) and ENAS-FERNet (ResBlock) can achieve lower time consumption, improved search efficiency and better performance than existing fully automated automatic search methods. As shown in Table III, our algorithm achieves the best results in a variety of situations. With noise rates of 10%, our ENAS-FERNet achieved the best results and established new SOTA result, specifically, ENAS-FERNet (ConvBlock) and ENAS-FERNet (ResBlock) achieved 0.52%, and 0.94% better results than the second best performing algorithm. Thus, the above experiment reflects the fact that ENAS-FERNet is very robust against uncertainty.

The performance of ENAS-FERNet during the evolutionary process: As we have argued throughout the article, the depth of a deep neural network seems to play an important role in the performance of the network. We have defined multiple mutation operators, as described in Section IV. These mutations allow to automatic modify the depth of the network. To better understand the effects of these mutations on the accuracy of ENAS-FERNet and the depth of the network on the RAF-DB dataset, we present the results in Figs. 5 and 6, respectively. The blue dotted line, with square markers, represents ENAS-FERNet (ResBlock), while the red dotted line with star symbols represents ENAS-FERNet (ConvBlock). Specifically, the blue dotted line reaches

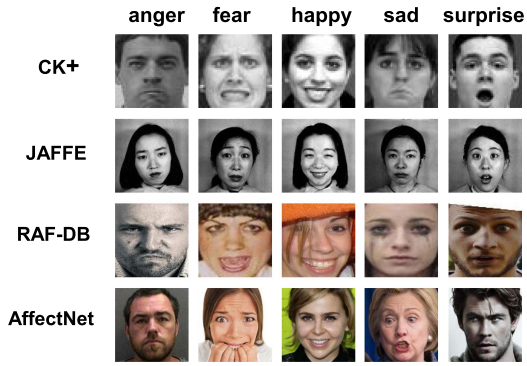

Fig. 4. For each dataset, five images are taken as examples, which correspond to five expressions: angry, fear, happy, sad, and surprise.

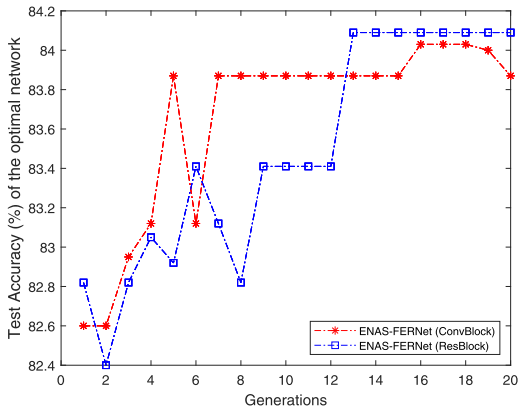

Fig. 5. Test accuracy of the optimal network architecture in populations on the RAF-DB dataset varies with the growth of population generations.

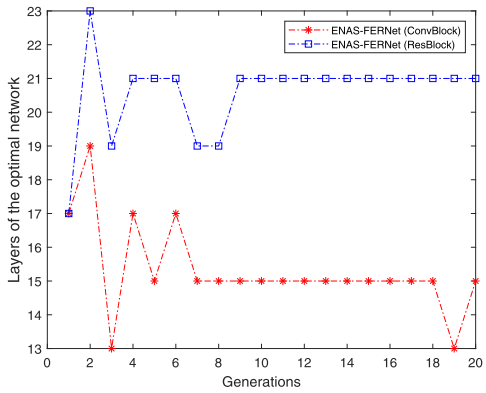

Fig. 6. Depth of the optimal network architecture in the population on the RAF-DB dataset varies with the growth of population generations.

global convergence faster and has a higher correct classification rate than the red dotted line, as shown in Fig. 5, which demonstrates that adding the skip connection into the designed Block can achieve better results. Of course, even though the use of skip connections improves the accuracy to some extent, it may lead to a tendency for deeper network architectures, as discussed below. The red dotted line in Fig. 6 shows that the final well-performing architecture with depth of 15 layers, while the depth of initial architecture that starts to perform well is greater than 15 layers. This is where the "depth-decreasing operator" of the proposed mutation operation comes into play, reducing the depth of the architecture throughout the evolutionary process and eventually converging to the optimal network depth. In contrast, the blue dot in Fig. 6 shows that the depth of the architecture with good performance in the initial population is 17, which is smaller than the depth of the architecture with good performance at the end of the evolution, which is 21. In this regard, the "depth-increasing operator" in the proposed mutation operation can be used to increase the depth of the architecture during the evolutionary process, which will eventually converge to the optimal network depth.

A comprehensive comparative analysis about ENAS-FERNet: As mentioned in the introduction, it is a very tedious and error-prone process for people to design neural network architectures for FER manually [3], and for this reason NAS methods have emerged for semi-automated and fully automated design of neural network architectures [24], [30]. After experimental validation on several challenging and noisy datasets, ENAS-FERNet is compared with the manually designed, semi-automated, and fully automated methods. The comparative analysis is shown in Table VI. From Table VI, it can be seen that ENAS-FERNet is higher than the existing manual methods on the CK+, AffectNet, RAF-DB, RAF-DB (10%), and RAF-DB (20%) datasets, ranging from 0.20% to 11.70%, respectively, compared to the manually designed neural network architectures, etc. ENAS-FERNet outperforms the existing automated methods on the CK+, JAFFE, AffectNet, RAF-DB, and RAF-DB (10%) datasets by 0.15% to 4.60%, respectively, and so on. This well demonstrates that ENAS-FERNet has good generalization capability and thus solves the problem of poor generalization capability of existing manual and automated methods. From the perspective of time consumption, on the one hand, ENAS-FERNet combined with variable-length encoding strategy and the cache system can be comparable and competitive with gradient-based methods in terms of speed, for example, DARTS takes 19.32 hours and 28 hours on CK+ and RAF-DB (20%) datasets, respectively, and ENAS-FERNet (ConvBlock) takes 13.33 hours and, 29 hours, respectively. And another advantage of ENAS-FERNet over DARTS is that ENAS-FERNet can fully automate the search, while DARTS requires manual assistance. On the other hand, ENAS-FERNet has a significant speedup compared with the genetic programming-based method CGP, for example, CGP takes 124 hours, 619 hours, 125 hours and 124 hours for the four datasets RAF-DB, AffectNet, RAF-DB (10%) and RAF-DB (20%), respectively, in contrast ENAS-FERNet (ResBlock) takes 49 hours, 170 hours, 62 hours and 72 hours on these four datasets, which is a significant improvement and can effectively improve the search efficiency. In addition, we find that most of the network architectures learned by ENAS-FERNet (ConvBlock) consist of a stack of three ConvBlocks combined with a pooling layer. Similarly, most of the network architectures learned by ENAS-FERNet (ResBlock) consist of a stack of two ResBlocks combined with a pooling layer. These results show that the above two stacking forms of these basic modules are beneficial to make the model perform better.

VI. Conclusion

In this article, we propose a generic automated FER network ENAS-FERNet, which can be divided into two models ENAS-FERNet (ConvBlock) and ENAS-FERNet (ResBlock) depending on the encoding modules. ENAS-FERNet is able to encode the neural network architecture as the diversity of individuals in the population, and then update the population by the proposed crossover and mutation operations, where four types of mutation operations enable ENAS-FERNet to explore better individuals, and finally obtain a network architecture with good performance after continuous evolution. Furthermore, ENAS-FERNet is validated on two laboratory-controlled datasets, two in-the-wild datasets and two datasets with synthesized noise for validation and analysis. The comprehensive analysis and results show that the proposed ENAS-FERNet has good generalization capability, which in turn solves the problem of poor generalization capability of existing automated methods. ENAS-FERNet combined with the designed caching system can effectively improve the search efficiency of ENAS-FERNet and can well solve the problem of high time consumption of existing fully automated methods. In the future, there are more challenging applications for FER where the pose changes or the dataset is occluded, and we will explore automated FER in these areas.

References

[1] A. Dapogny, K. Bailly, and S. Dubuisson, "Dynamic pose-robust facial expression recognition by multi-view pairwise conditional random forests," *IEEE Trans. Affect. Comput.*, vol. 10, no. 2, pp. 167–181, Apr.–Jun. 2019.

[2] D. Liu, X. Ouyang, S. Xu, P. Zhou, K. He, and S. Wen, "SAANet: Siamese action-units attention network for improving dynamic facial expression recognition," *Neurocomputing*, vol. 413, pp. 145–157, 2020.

[3] S. Minaee, M. Minaei, and A. Abdolrashidi, "Deep-emotion: Facial expression recognition using attentional convolutional network," *Sensors*, vol. 21, no. 9, 2021, Art. no. 3046.

[4] B. Zoph and Q. V. Le, "Neural architecture search with reinforcement learning," 2016, in *Proc. Int. Conf. Learn. Representations*, 2016. [Online]. Available: https://openreview.net/pdf?id=r1Ue8Hcxg

[5] 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, 2009.

[6] S. Li, W. Deng, and J. Du, "Reliable crowdsourcing and deep locality-preserving learning for expression recognition in the wild," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 2852–2861.

[7] A. Mollahosseini, B. Hasani, and M. H. Mahoor, "AffectNet: A database for facial expression, valence, and arousal computing in the wild," *IEEE Trans. Affect. Comput.*, vol. 10, no. 1, pp. 18–31, Jan.–Mar. 2019.

[8] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[9] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," *Commun. ACM*, vol. 60, no. 6, pp. 84–90, 2017.

[10] D. Meng, X. Peng, K. Wang, and Y. Qiao, "Frame attention networks for facial expression recognition in videos," in *Proc. IEEE Int. Conf. Image Process.*, 2019, pp. 3866–3870.

[11] K. Wang, X. Peng, J. Yang, D. Meng, and Y. Qiao, "Region attention networks for pose and occlusion robust facial expression recognition," *IEEE Trans. Image Process.*, vol. 29, pp. 4057–4069, 2020.

[12] K. Wang, X. Peng, J. Yang, S. Lu, and Y. Qiao, "Suppressing uncertainties for large-scale facial expression recognition," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2020, pp. 6897–6906.

[13] Q. Lin, Z. Fang, Y. Chen, K. C. Tan, and Y. Li, "Evolutionary architectural search for generative adversarial networks," *IEEE Trans. Emerg. Topics Comput. Intell.*, vol. 6, no. 4, pp. 783–794, Aug. 2022.

[14] P. Ren et al., "A comprehensive survey of neural architecture search: Challenges and solutions," *ACM Comput. Surv.*, vol. 54, no. 4, pp. 1–34, 2021.

[15] S. Zagoruyko and N. Komodakis, "Wide residual networks," in *Proc. Brit. Mach. Vis. Conf.*, Sep. 2016, pp. 87.1–87.12.

[16] T. Back, *Evolutionary Algorithms in Theory and Practice: Evolution Strategies, Evolutionary Programming, Genetic Algorithms*. London, U.K.: Oxford Univ. Press, 1996.

[17] R. Negrinho and G. Gordon, "DeepArchitect: Automatically designing and training deep architectures," 2017, *arXiv:1704.08792*.

[18] J. Bergstra and Y. Bengio, "Random search for hyper-parameter optimization," *J. Mach. Learn. Res.*, vol. 13, no. 2, pp. 281–305, 2012.

[19] C. Liu et al., "Progressive neural architecture search," in *Proc. Eur. Conf. Comput. Vis.*, 2018, pp. 19–34.

[20] B. Baker, O. Gupta, N. Naik, and R. Raskar, "Designing neural network architectures using reinforcement learning," in *Proc. Int. Conf. Learn. Representations*, 2016. [Online]. Available: https://openreview.net/pdf?id=S1c2cvqee

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

[22] Z. Zhong, J. Yan, and C.-L. Liu, "Practical network blocks design with Q-learning," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 2423–2432.

[23] B. Zoph, V. Vasudevan, J. Shlens, and Q. V. Le, "Learning transferable architectures for scalable image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 8697–8710.

[24] H. Liu, K. Simonyan, and Y. Yang, "DARTS: Differentiable architecture search," in *Proc. Int. Conf. Learn. Representations*, 2018. [Online]. Available: https://openreview.net/pdf?id=S1eYHoC5FX

[25] R. Shin, C. Packer, and D. Song, "Differentiable neural network architecture search," 2018. [Online]. Available: https://openreview.net/forum?id=BJ-MRKkwG

[26] X. Chen, L. Xie, J. Wu, and Q. Tian, "Progressive differentiable architecture search: Bridging the depth gap between search and evaluation," in *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, 2019, pp. 1294–1303.

[27] Y. Xu et al., "PC-DARTS: Partial channel connections for memory-efficient architecture search," in *Proc. Int. Conf. Learn. Representations*, 2020. [Online]. Available: https://openreview.net/pdf?id=BJlS634tPr

[28] E. Real et al., "Large-scale evolution of image classifiers," in *Proc. Int. Conf. Mach. Learn.*, 2017, pp. 2902–2911.

[29] L. Xie and A. Yuille, "Genetic CNN," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2017, pp. 1379–1388.

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

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

[32] Y. Liu, Y. Sun, B. Xue, M. Zhang, G. G. Yen, and K. C. Tan, "A survey on evolutionary neural architecture search," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 34, no. 2, pp. 550–570, Feb. 2023.

[33] Y. Sun, B. Xue, M. Zhang, and G. G. Yen, "Evolving deep convolutional neural networks for image classification," *IEEE Trans. Evol. Comput.*, vol. 24, no. 2, pp. 394–407, Apr. 2020.

[34] T. Elsken, J. H. Metzen, and F. Hutter, "Neural architecture search: A survey," *J. Mach. Learn. Res.*, vol. 20, no. 1, pp. 1997–2017, 2019.

[35] Y. Sun, B. Xue, M. Zhang, G. G. Yen, and J. Lv, "Automatically designing CNN architectures using the genetic algorithm for image classification," *IEEE Trans. Cybern.*, vol. 50, no. 9, pp. 3840–3854, Sep. 2020.

[36] X. Xie, X. Song, Z. Lv, G. G. Yen, W. Ding, and Y. Sun, "Efficient evaluation methods for neural architecture search: A survey," 2023, *arXiv:2301.05919*.

[37] D. Zhou et al., "EcoNAS: Finding proxies for economical neural architecture search," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2020, pp. 11396–11404.

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

[39] B. Evans, H. Al-Sahaf, B. Xue, and M. Zhang, "Evolutionary deep learning: A genetic programming approach to image classification," in *Proc. IEEE Congr. Evol. Comput.*, 2018, pp. 1–6.

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

[41] J. J. Grefenstette, "Genetic algorithms and machine learning," in *Proc. 6th Annu. Conf. Comput. Learn. Theory*, 1993, pp. 3–4.

[42] D. G. R. Kola and S. K. Samayamantula, "A novel approach for facial expression recognition using local binary pattern with adaptive window," *Multimedia Tools Appl.*, vol. 80, no. 2, pp. 2243–2262, 2021.
[43] H. Ding, S. K. Zhou, and R. Chellappa, "FaceNet2ExpNet: Regularizing a deep face recognition net for expression recognition," in *Proc. IEEE 12th Int. Conf. Autom. Face Gesture Recognit.*, 2017, pp. 118–126.
[44] M. Pourmirzaei, G. A. Montazer, and F. Esmaili, "Using self-supervised auxiliary tasks to improve fine-grained facial representation," 2021, *arXiv:2105.06421*.
[45] S. Aghera, H. Gajera, and S. K. Mitra, "MnasNet based lightweight CNN for facial expression recognition," in *Proc. IEEE Int. Symp. Sustain. Energy, Signal Process. Cyber Secur.*, 2020, pp. 1–6.
[46] S. Li et al., "Auto-FERNet: A facial expression recognition network with architecture search," *IEEE Trans. Netw. Sci. Eng.*, vol. 8, no. 3, pp. 2213–2222, Jul.–Sep. 2021.
[47] S. Katoch, S. S. Chauhan, and V. Kumar, "A review on genetic algorithm: Past, present, and future," *Multimedia Tools Appl.*, vol. 80, no. 5, pp. 8091–8126, 2021.
[48] B. Fox and M. McMahon, "Genetic Operators for Sequencing Problems," in *Foundations of Genetic Algorithms*. Amsterdam, The Netherlands: Elsevier, 1991, pp. 284–300.
[49] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, Nov. 1997.
[50] F. A. Gers, J. Schmidhuber, and F. Cummins, "Learning to forget: Continual prediction with LSTM," *Neural Computation*, vol. 12, no. 10, pp. 2451–2471, Oct. 2000.
[51] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2016, pp. 770–778.
[52] S. Fujino, N. Mori, and K. Matsumoto, "Deep convolutional networks for human sketches by means of the evolutionary deep learning," in *Proc. IEEE Joint 17th World Congr. Int. Fuzzy Syst. Assoc. 9th Int. Conf. Soft Comput. Intell. Syst.*, 2017, pp. 1–5.
[53] P. Lucey, J. F. Cohn, T. Kanade, J. Saragih, Z. Ambadar, and I. Matthews, "The extended Cohn-Kanade dataset (CK+) : A complete dataset for action unit and emotion-specified expression," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.-Workshops*, 2010, pp. 94–101.
[54] M. J. Lyons, M. Kamachi, and J. Gyoba, "Coding facial expressions with Gabor wavelets (IVC special issue)," 2020, *arXiv:2009.05938*.
[55] M. J. Lyons, "Excavating AI" re-excavated: Debunking a fallacious account of the JAFFE dataset," 2021, *arXiv:2107.13998*.
[56] H. Zhao, Q. Liu, and Y. Yang, "Transfer learning with ensemble of multiple feature representations," in *Proc. IEEE 16th Int. Conf. Softw. Eng. Res., Manage. Appl.*, 2018, pp. 54–61.
[57] Y. Yaddaden, M. Adda, A. Bouzouane, S. Gaboury, and B. Bouchard, "User action and facial expression recognition for error detection system in an ambient assisted environment," *Expert Syst. Appl.*, vol. 112, pp. 173–189, 2018.
[58] Y. Yaddaden, M. Adda, and A. Bouzouane, "Facial expression recognition using locally linear embedding with LBP and hog descriptors," in *Proc. IEEE 2nd Int. Workshop Hum.-Centric Smart Environ. Health Well-Being*, 2021, pp. 221–226.
[59] M. Aouayeb, W. Hamidouche, C. Soladie, K. Kpalma, and R. Seguier, "Learning vision transformer with squeeze and excitation for facial expression recognition," 2021, *arXiv:2107.03107*.
[60] S. Guo et al., "CurriculumNet: Weakly supervised learning from large-scale web images," in *Proc. Eur. Conf. Comput. Vis.*, 2018, pp. 135–150.
[61] W. Zhang, Y. Wang, and Y. Qiao, "MetaCleaner: Learning to hallucinate clean representations for noisy-labeled visual recognition," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 7373–7382.
[62] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," 2014, *arXiv:1409.1556*.
[63] Z. Zhao, Q. Liu, and F. Zhou, "Robust lightweight facial expression recognition network with label distribution training," in *Proc. AAAI Conf. Artif. Intell.*, 2021, pp. 3510–3519.

Shuchao Deng received the B.S. degree in computer science in 2022 from Sichuan University, Chengdu, China, where he is currently working toward the M.E. degree in computer science. His research interests include evolutionary neural architecture search and physics-informed neural networks.

Zeqiong Lv (Student Member, IEEE) received the M.E. degree in computer science from Xihua University, Chengdu, China, in 2021. She is currently working toward the Ph.D. degree in computer science from Sichuan University, Chengdu. Her research interests include evolutionary computation, neural networks, and theoretical analysis of evolutionary algorithms.

Edgar Galván (Member, IEEE) received the Ph.D. degree in computer science from University of Essex, Colchester, U.K. He is currently a Senior Researcher with the Department of Computer Science, National University of Ireland Maynooth, Maynooth, Ireland. Previously, he was a Marie Curie Fellow with INRIA Paris-Saclay France, Research Fellow with the School of Computer Science and Statistics, Trinity College Dublin, Visiting Researcher with the School of Computer Science and Engineering, Essex University, and Postdoctoral Researcher with the Complex and Adaptive System Laboratory, University College Dublin. He has coauthored more than 80 scientific papers and book chapters. His research interests include applications to combinatorial optimisation, neural networks, neuroevolution, games, and software engineering. He has independently ranked as one of the all-time top 1% researchers of genetic programming, according to University College London, London, U.K.

Yanan Sun (Member, IEEE) received the Ph.D. degree in computer science from Sichuan University, Chengdu, China, in 2017. He is currently a Professor with the College of Computer Science, Sichuan University. His research interests include evolutionary computation, neural networks, and their applications on neural architecture search. He designed the indicator of "GPU Day", which has been widely used among the community of neural architecture search. He was ranked as World's Top 2% Scientists 2021, collectively released by Stanford University and Springer. He is an Associate Editor for IEEE TRANSACTIONS ON EVOLUTIONARY COMPUTATION.