A Survey on Evolutionary Neural Architecture Search

Yuqiao Liu, *Student Member, IEEE,* Yanan Sun, *Member, IEEE,* Bing Xue, *Member, IEEE,* Mengjie Zhang, *Fellow, IEEE,* and Gary G. Yen, *Fellow, IEEE*

***Abstract*—Deep Neural Networks (DNNs) have achieved great success in many applications, such as image classification, natural language processing and speech recognition. The architectures of DNNs have been proved to play a crucial role in its performance. However, mamually designing architectures for different tasks is a difficult and time-consuming process of trial and error. Neural Architecture Search (NAS), which received great attention in recent years, can design the architecture automatically. Among different kinds of NAS methods, Evolutionary Computation (EC) based NAS methods have recently gained much attention and success. Unfortunately, there has not yet been a comprehensive summary of the EC-based methods. This paper reviews 100+ papers of EC-based NAS methods in light of the common process. Four steps of the process have been covered in this paper including encoding space determination, population initialization, population updating and evaluation. Furthermore, current challenges and issues are also discussed to identify future research in this field.**

***Index Terms*—Evolutionary Neural Architecture Search, Evolutionary Computation, deep learning, image classification.**

I. Introduction

DEEP Neural Networks (DNNs), as the cornerstone of deep learning [1], have demonstrated their great success in diverse real-world applications, including image classification [2], [3], natural language processing [4], speech recognition [5], to name a few. The promising performance of DNNs has been widely recognized largely due to their deep architectures, being able to learn meaningful features directly from the raw data almost without any explicit feature engineering. Generally, the performance of DNNs depends on two aspects: their architectures and the associated weights. Only when both achieve the optimal status simultaneously, the performance of the DNNs could be satisfactory. The optimal weights are often obtained through the local learning process: using a continuous loss function to measure the difference between the real output and the desired output, and then the gradient-based algorithms are often used to minimize the loss. When the termination satisfied the condition which is commonly a maximal iteration number, the weights obtained at that point are used as the optimum. Such kind of learning process has been very popular largely owe to its effectiveness in practice, and has become the dominant technique for weight optimization [6], although the gradient-based optimization algorithm is principally local-search [7]. However, obtaining the architectures cannot be directly formulated by a continuous function, and there is even no explicit function to measure the process for finding optimal architectures.

Yuqiao Liu and Yanan Sun are with the College of Computer Science, Sichuan University, China.

Bing Xue and Mengjie Zhang are with the School of Engineering and Computer Science, Victoria University of Wellington, Wellington, New Zealand.

Gary G. Yen is with the School of Electrical and Computer Engineering, Oklahoma State University, Stillwater, OK, USA.

To this end, there has been a long time that the promising architectures of DNNs are manually designed with rich expertise. This can be evidenced from the state-of-the-arts, such as VGG [8], ResNet [2] and DenseNet [3]. These promising Convolutional Neural Network (CNN) models are all manually designed by the researchers with rich knowledge in both neural networks and image processing. However, in practice, most end users lack such kind of rich knowledge. Moreover, the DNN architectures are often problem-dependent. If the distribution of the data is changed, the architectures must be redesigned accordingly. Neural Architecture Search (NAS), which aims to automate the architecture designs of **deep** neural networks, is recognized as an effective and efficient way to solve the limitations aforementioned.

Mathematically, the NAS is generally modeled by an optimization problem formulated by Equation (1):

$$\begin{cases} \arg\min_{A} = \mathcal{L}(A, \mathcal{D}_{train}, \mathcal{D}_{valid}) \\ \quad s.t. \quad A \in \mathcal{A} \end{cases} \tag{1}$$

where $\mathcal{A}$ denotes the search space of the neural architectures, $\mathcal{L}(\cdot)$ measures the performance of the architecture A on the validation dataset $\mathcal{D}_{valid}$ after being trained on the training dataset $\mathcal{D}_{train}$. The $\mathcal{L}(\cdot)$ is always non-convex and non-differential [9]. In principle, NAS is a complex optimization problem experiencing several challenges, e.g. complex constraints, discrete representations, bi-level structures, computationally expensive characteristics and multiple conflicting objectives. NAS algorithms refer to the optimization algorithms which are specifically designed to effectively and efficiently solve the problem represented by Equation (1). This initial work of NAS algorithms is commonly viewed as the work in [10], which was proposed by Google. The pre-print version of this work was firstly released in Arvix in 2016, and then was formally accepted for publication by the International Conference on Learning Representations (ICLR) in 2017. Since then, a vast number of researchers have invested tremendous effort on novel NAS algorithms.

Based on the optimizer employed, existing NAS algorithms can be generally classified into three different categories: Reinforcement Learning (RL) [11] based NAS algorithms, gradient based NAS algorithms and Evolutionary Computation (EC) [12] based NAS algorithms (ENAS). Specifically, the

RL based algorithms often require thousands of Graphics Processing Cards (GPUs) performing several days even on the median-scale dataset, such as the CIFAR10 image classification benchmark dataset [13]. The gradient based algorithms are more efficient than the RL based algorithms. However, they often find the ill-conditioned architectures due to the incorporated relation for adapting to the gradient based optimization. Unfortunately, the relation has not been mathematically proven. In addition, the gradient based algorithms require to construct a supernet in advance, which also highly requires expertise. The ENAS algorithms solve the NAS problems by exploiting EC techniques. Specifically, EC is a kind of population-based computational paradigm, simulating the evolution of species or the behavior of the population in nature, to solve challenging optimization problems. In particular, genetic algorithms (GA) [14], genetic programming (GP) [15], and particle swarm optimization (PSO) [16] are the widely employed EC methods in practice. Owing to the promising characteristics of EC methods in insensitiveness to the local minimal and no requirements to gradient information, EC has been widely applied to solve non-convex optimization problems [17], even when the mathematical form of the objective function is not available [18].

In fact, the EC methods have been frequently and widely used more than twenty years ago, searching for not only the optimal neural architectures but also the weights, which is also termed as "neuroevolution". The major differences between ENAS and neuroevolution lie in two aspects. Firstly, neuroevolution often search for both the neural architectures and the optimal weight values, while ENAS focuses mainly on the architectures and the optimal weight values are obtained by using the gradient-based algorithms.[1] Secondly, the neuroevolution commonly applies to small-scale and median-scale neuron networks, while ENAS widely works on DNNs, such as the Convolutional Neural Networks (CNNs) [22], [23] and deep stacked autoencoders [24], which are the building blocks of deep learning technique [1]. Generally, the initial work of ENAS is viewed as the LargeEvo algorithm [22] which was proposed by Google who released its early version in March of 2017 in Arxiv, and then this paper got accepted by the 34th International Conference on Machine Learning in June of 2017. The LargeEvo algorithm employed a GA to search for the best architecture of a CNN, and the experimental results on CIFAR-10 and CIFAR-100 [13] have demonstrated its effectiveness. Since then, a large number of ENAS algorithms have been proposed. Fig. 1 shows the number of "submissions" [2] focusing on the ENAS algorithms, from 2017 to the April of 2020 when we make this survey paper. As can be seen from this figure, from 2017 to 2019, the number of submissions grows with multiple scales, and in the first quarter of 2020, the submission is almost the same number as those in the whole year of 2018.

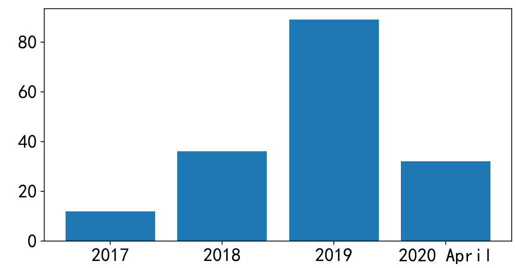

Fig. 1. The number of "submissions" refers to the works of evolutionary neural architecture search. The data is from Google Scholar with the keywords of "evolutionary" OR "genetic algorithm" OR "particle swarm optimization" OR "PSO" OR "genetic programming" AND "architecture search" OR "architecture design" OR "CNN" OR "deep learning" OR "deep neural network" and the literature on Neural Architecture Search collected from the AutoML.org website by the end of April 20, 2020. With these initially collected data, we then have carefully checked each manuscript to make its scope accurately within the evolutionary neural architecture search.

A great number of related submissions have been made available publicly, but there is no comprehensive survey of the literature on ENAS algorithms. Although recent reviews on NAS have been made in [18], [25], [26] and [27], they mainly focus on all kinds of methods of NAS, and make a macro summery on NAS, instead of concentrating on the ENAS algorithms. To be specific, Elsken*et al.* in [25] divided NAS into three stages: Search Space, Search Strategy, and Performance Estimation Strategy. Similarly, Wistuba *et al.* in [26] also followed these three stages with an additional review about the multiple objectives NAS. Darwish *et al.* in [18] made a summary of Swarm Intelligence (SI) and Evolutionary Algorithm (EA) approaches for deep learning, with the focuses on both NAS and other hyperparameter optimization. Stanley *et al.* in [27] went through a review of neuroevolution, aiming at revealing the wight optimization rather than the architecture. In addition, most of the references in these surveys aforementioned are pre-2019 and do not largely involve the papers in the past two years when most ENAS works are published. This paper makes a survey involving a large number of ENAS papers, with the expectation to inspire some new ideas for enhancing the development of ENAS. To make the readers more easily focusing on the technical part of this survey, we also follow the three stages for the introduction to ENAS algorithms, which has been widely adopted by existing NAS survey papers [18], [25], [26], but with essential modifications made to specifically suit ENAS algorithms.

The reminder of this paper is organized as follows. The background of ENAS is provided in Section II. Section III documents different encoding space, initial space and search space. In Section IV, the encoding strategy and architecture representation are introduced. Section V summarizes the whole process of population updating, including the evolutionary operators and the selection strategy. Section VI shows several ways to speed up the evolution. Section VII presents the applications of ENAS algorithms. Section VIII discusses the challenges and prospects, and Section IX is for the conclusion.

[1]Please note that we still categorize some existing algorithms as the ENAS algorithm, such as API [19], EvoCNN [20] and EvoDeep [21], although they also concern the weights. This is because the optimal weight values of the DNNs searched by them are still obtained by the gradient-based algorithms. They only searched for the best weight initialization values or the best weight initialization method of the DNNs.

[2]These "submissions" include the ones which have been accepted for publication after the peer-review process, and also the ones which are only available on the Arxiv website without the peer-review process.

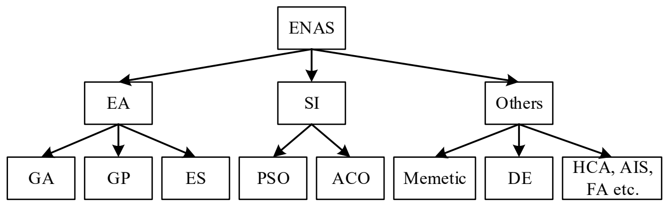

Fig. 2. The categories of ENAS from EC methods regarding the search strategies.

II. Background

ENAS is a process of searching for the optimal deep neural architectures by using evolutionary search strategy from different EC paradigms. In this section, we will first introduce the categories of EC paradigms based on their search strategies and the categories of neural architectures in Subsections II-A and II-B, respectively. Then, the unified flow chart of ENAS is presented in Subsection II-C.

A. Evolutionary Search Strategy

ENAS is distinguished from other NAS by its optimization approach, i.e. the EC methods. While the optimization approach can be further subdivided based on the search strategy. Fig. 2 provides such an illustration from three aspects: EA, swarm intelligence (SI), and others, and a more detailed statistics will be shown in Table III. Specifically, the EA-based methods account for the majority of existing ENAS algorithms, and the GA-based methods take a large part of EA-based methods. The other categories of EC methods are also an important part for realizing ENAS algorithms, such as GP, Evolutionary Strategy (ES), PSO, Ant Colony Optimization (ACO) [28], Differential Evolution (DE) [29], Firefly Algorithm (FA) [30], and etc. Note that, we classify the Hill-Climbing Algorithm (HCA) into EC paradigm, because it can be regarded as a EA with a very simple selection mechanism and no crossover operation [31]. It has been well known as a widely used local search algorithm in memetic algorithm [32].

B. Common Neural Networks in ENAS

As mentioned above, the target of ENAS is mainly about the DNNs, which can be divided into five different categories: CNN, Deep Belief Networks (DBN), Stacked Auto-Encoder (SAE), Recurrent Neural Network (RNN) and others. The brief fact can be seen from Fig. 3, and a detailed illustration will show in Table III.

In reality, most ENAS methods are proposed for searching for the optimal CNN architectures. This is because many hand-crafted CNNs, such as VGG [8], ResNet [2] and DenseNet [3], have shown their superiority in handling the image classification task which is the most successful applications in the field. Generally, the CNN architecture is composed of convolutional layer, pooling layer and fully-connected layer, and a common example of CNNs is shown in Fig. 4 (a). The optimization in CNN architecture is mainly divided into three aspects: the hyparameters of each layer [33], the depth of the architecture [20] and the connections between layers [34]. Majority of the ENAS methods consider the above three aspects collectively [22], [35].

DBN [36] is made up by stacking multiple Restricted Boltzmann Machines (RBMs), an example of which can be seen in Fig. 4 (b). RBMs have connections only between layers, while without inter-layer connection. Meanwhile, the RBMs allow the DBN to be trained in an unsupervised way first to obtain a good weight initialization. There are two hyparameters needed to be optimized in DBN: the number of neurons in each layer [37] and the number of layer [38], [39].

A SAE is also constructed by stacking multiple AEs. The aim of AE is to learn the meaningful representations from the raw input data by restoring it input data as its objective [40]. Generally, the AE is composed of two symmetrical components: the encoder and the decoder, and an example including two parts is shown in Fig. 4 (c). A part of ENAS methods only encode the encoder into evolution [24], [41], because the decoder part is symmetric with the encoder and can be generated automatically. While some other ENAS methods optimize the hyparameters of encoder and decoder separately [42], [43].

The most significant difference between RNN and other neural networks introduced above is its recurrent connection in the architecture. Fig. 4 (d) shows the time-expanded structure of a RNN, where the value of current hidden layer h^t is influenced by the value at its previous time slot h^{t-1} and the value of its previous layer. Because these layers are reused, all the weights (i.e. U, W and V) in the figure are constant. Different from focusing on the number of neurons and the number of layers in feedforward neural network, some ENAS methods concern about how many times the RNN is unfold [44], [45]. In addition, there remains other neural networks like typical DNNs which are made up of only fully-connected layers, where the connections are formed by all the neurons in the two adjacent layers.

When the ENAS algorithms are applied to these neural networks, the goal is often to find the best architecture-related parameters. Specifically, for CNNs, the number of convolutional layers, pooling layer, fully-connected layers, and parameters related to these layers (such as the kernel size of the convolutional layers, the pooling type, and the number of neurons for fully-connected layers, and so on) as well as their connection situation (such as the dense connection and the skip connection). For DBN and SAE, the number of their building blocks, i.e., the RBM for DBN and the AE for SAE, and the number of each layer, are the commonly architecture-related parameters to be optimized. For RNN, in addition to the architecture-related parameters mentioned above, the number of time slot is often also an important parameter to be optimized by ENAS algorithms. For the traditional DNNs, the ENAS algorithms often concern on the neuron number of each layer. In addition, some ENAS algorithms also concern the aspect to the weights, such as the weight initialization method and the weigh initial values. The details of these EANS algorithms will be documented in the following.

Table I displays the detail of the common parameters optimized by ENAS methods in different neural networks especially CNN which is the most popular in ENAS.

TABLE I
COMMON PARAMETERS OPTIMIZED IN DIFFERENT NEURAL NETWORK.

	Parameters	
CNN	global parameters	number of layers, connections between layers
	convolution layer	filter size (width and height), stride size (width and height), featrue map size, convolution type, standard deviation and mean value of the filter elements
	pooling layer	filter size (width and height), stride size (width and height), pooling type
	fully-connected layer	number of neurons, standard deviation and mean value of weights
DBN, AE	number of hidden layers, neurons per layer	
RNN	number of hidden layers, neurons per layer, number of time slot	

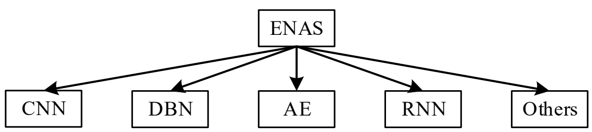

Fig. 3. The categories of ENAS from neural network perspective.

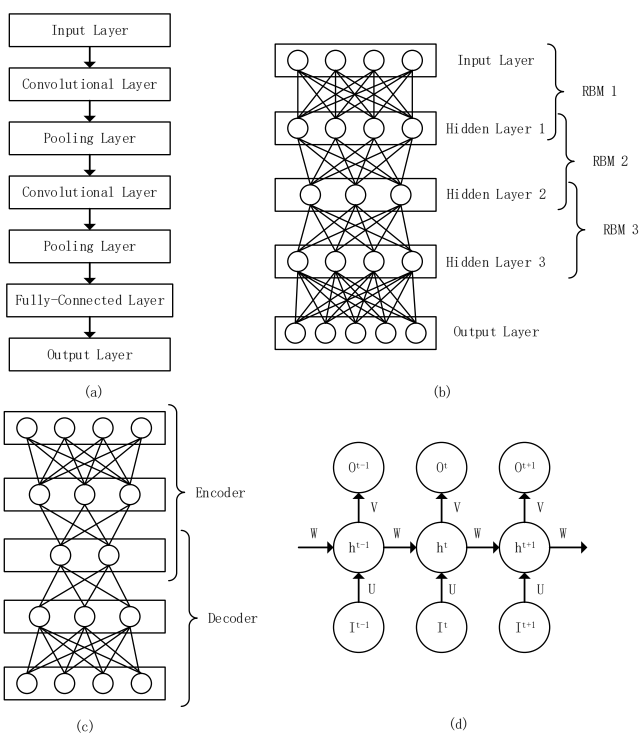

Fig. 4. Examples of different neural architectures in ENAS. (a) CNN. (b) DBN. (c) AE. (d) RNN.

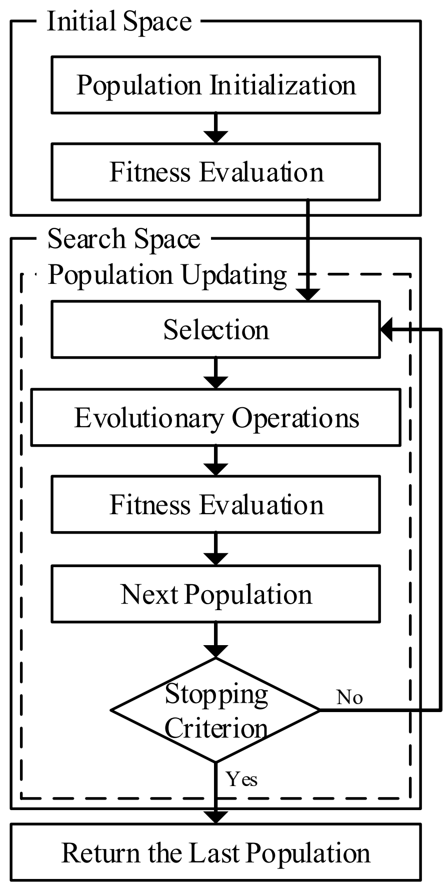

Fig. 5. The flowchart of a common ENAS algorithm.

C. *Flow Chart of ENAS*

An illustration of the flow chart of an EANS algorithm is shown in Fig. 5. The evolution will take place in the initial space and the search space sequentially, and these two spaces are introduced in Section III. First of all, a population is initialized within the initial space defined in advance. Particularly, each individual in the population represents a solution for the NAS, i.e., a DNN architecture. Each architecture needs to be encoded as an individual before it joins the population. Second, the fitness of the generated individuals is evaluated. Note that there are two fitness evaluation stages shown in Fig. 5, which employ the same evaluation criterion. Thirdly, after the fitness evaluation of the initial population, the whole population starts to the evolutionary process within the search space, which is shown by the dashed box in Fig. 5. In the evolutionary process, the population is updated by the selection and the evolutionary operators in each evolutionary step, until the stopping criterion is met. Please note that the selection stage is not necessary for some other EC paradigms like SI. Finally, a population that has finished the evolution is obtained.

III. ENCODING SPACE

Encoding space contains all the valid encoding individuals. The encoding space can be divided into three categories according to the basic units they adopt: the layer-based, the block-based and the cell-based. Furthermore, some ENAS methods do not care about the configuration of the basic unit, while they only concern the connections between units. Therefore, they are classified into the fourth class of encoding space: the topology-based.

In addition, the constraints on the encoding space are important, because the constraints represent the human intervention which restricts the encoding space and lighten the burden of evolutionary process. A method with a mass of constraints can obtain a promising architecture easily but prevent to design a novel architecture. Furthermore, the different size of search space would greatly affect the efficiency of evolution. We can not evaluate the effectiveness of an ENAS method without its constraints on the search space, because one extreme case is that all the individuals in the search space are well performed. In this case, an excellent individual can be searched even if do no selection. We will introduce some typical constraints in ENAS next.

Table II shows different kinds of encoding space and the constraints. Vertical classification is based on the categories of the basic units searched, and horizontal classification is based on the constraints. We will introduce the encoding space from this two kinds of classification.

As introduced above, the encoding space can be divided into four categories: layer-based, block-based, cell-based and topology-based. And we will introduce them in this order.

The layer-based denotes the basic units in the encoding space are primitive layers (such as convolution layers and fully-connected layers). The individuals in layer-based encoding space are much more delicate, because every low-level details can be considered. However, it may take more time to search for a delicate individual in such a large layer-based space.

To alleviate this, many researchers use blocks as the basic units, which are specific structure combinations of various types of layers. Many different kinds of blocks are presented: ResBlock [2], DenseBlock [3], ConvBlock (Conv2d + Batch-Normalization + Activation) [31] and InceptionBlock [140] etc. These blocks have promising performance and fewer parameters are needed to build the architecture. So it is easier to find a good architecture in the block-based encoding space. Some methods use these above blocks directly, such as [9], [75], and some other methods propose other blocks for their own purposes. Chen [107] *et al.* propose 8 blocks including ResBlock and InceptionBlock encoded in 3 bits string, and use Hamming distance to tell the similar blocks. Song [113] *et al.* propose three residual dense blocks to reduce the amount of computation due to the convolution operation of image super-resolution tasks.

Cell-based encoding space is similar to the block-based space, and it can be regarded as a special case in block-based space where all the blocks are the same. The methods choosing this space build the architecture by stacking repeated motifs. Chu *et al.* [128] divides the cell-based space into two irrelevant parts: the micro part contains the parameters of cells while the macro part defines the connections between different cells. The cell-based space greatly reduces the encoding space, but Frachon *et al.* [75] believes that there is no theoretical basis for that the cell-based space can help to get a good architecture.

In the last, the topology-based space does not care about the parameters or the structure of each unit (layer or block), and their only concern is the connections between units. One classical example is the One-Shot which treats all the architectures as different subgraphs of a supergraph [25]. Yang *et al.* [136] search the architecture by choosing different connections in the supergraph, the subgraph built by the chosen connections becomes an individual. Another typical case is pruning. Wu *et al.* [132] shallow VGGNet [8] on CIFAR-10, and the aim is to prune unimportant weight connections.

From the perspective of horizontal classification, the constraints on the encoding space mainly focus on three aspects: fixed depth, rich initialization and partial structure fixed.

The fixed depth means all the individuals in the population have the same depth. The fixed depth is a strong constraint and largely reduces the encoding space. Noting that the fixed depth is different from the *fixed-length encoding strategy* which will be introduced in Section IV. In Genetic CNN [34], for example, the *fixed-length encoding strategy* only limits the maximum depth. The node which is isolated (no connection) is simply ignored. By this way, the individuals can obtain different depth. The second constraint is rich initialization (i.e. the *well-designed space* in Section III-A), and this is also a strong constraint with a lot of manual experience. In this case, the initialized architectures are manually designed, which goes against the original intention of NAS. The partial structure fixed means the architecture is partially settled. For example, in [63] a max-pooling layer is added to the network after every set of four convolution layers.

In Table II, the relatively few constraints denotes the method has no restrictions on these above three aspects, but that is not to say there is absolutely no constraint. For example, in the classification task, fully connected layer is used as the last layer in some methods [20], [81], [126]. Furthermore, there are other constraints. Moreover, the maximum length is predefined in many methods including both *fixed-length encoding strategy* methods [34] and *variable-length encoding strategy* methods [20] resulted in preventing the method discovering a deeper architecture. Wang *et al.* [74] try to break the limit of maximum length by using a Gaussian distribution initialization mechanism. Irwin *et al.* [141] break the limit not by initialization, but using the evolutionary operators, the crossover and the mutation, to extend the depth to any size.

Generally, encoding space can be thought of as equivalent to search space which contains all the valid neural network architectures, because each encoded individual in encoding space corresponds to an architecture in search space. Furthermore, search space contains a subspace, the initial space. Fig. 6 shows the relationship between the three spaces. The search space is lager than the initial space when some manual constraints are added to the population initialization. When there are no such manual constrains, search space and initial space are equivalent. The initial space determines what kind of individuals may appear in the initial population, and the search space determines what kind of individuals may appear in the iteration of population updating, where an illustration is shown in Fig. 5.

TABLE II
DIFFERENT ENCODING SPACE AND THE CONSTRAINTS

	Fixed depth	Rich initialization	Partial structure fixed	Relatively few constraints
Layer-based	[33], [46]–[56]	[33], [57]–[61]	[19], [53], [62]–[72]	[20]–[22], [24], [38], [39], [41], [45], [73]–[105]
Block-based	[106]–[108]	[109]	[68], [109], [110]	[9], [23], [31], [75], [104], [111]–[121]
Cell-based			[122]–[124]	[35], [121], [125]–[131]
Topology-based		[132], [133]		[34], [134]–[139]

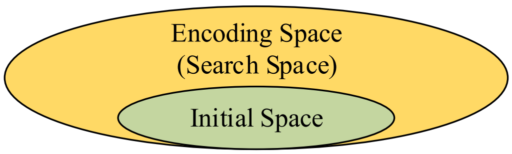

Fig. 6. The relationship between encoding space, search space and initial space.

A. Initial Space

In general, there are three types of architecture initialization: starting from trivial initial conditions [22], randomly initialization in encoding space [20] and staring from a well-designed architecture (also termed as rich initialization) [33]. These three types of initialization correspond to three different initial spaces: *trivial space*, *random space* and *well-designed space*.

The *trivial space* only contains few primitive layers. For example, Large-Scale Evolution [22] initializes the population in a *trivial space* where each individual constitutes just a single-layer model with no convolutions. Xie *et al.* [34] also did the experiment to prove that a *trivial space* can evolve to a competitive architecture. The reason using as little experience as possible is to justify the advantage of the evolution where some new can be discovered, and most of the discovery are different from the DNN architectures manually designed. On the contrary, the *well-designed space* contains the state-of-the-art architectures. In this way, a promising architecture can be obtained at the beginning of the evolution, whereas it can hardly evolve to other novel architectures. Actually, the majority ENAS methods adopting this initial space focus on improving the performance upon the well-designed architecture. For example, the architecture pruning aims at compressing DNNs by removing less important connections [133]. For *random space*, all the individuals in the initial population are randomly generated in the limited space, and many methods adopt *random space*, such as [9], [20], [66]. The aim of this type of initial space is also to reduce the intervention of human experience in the initial population.

B. Search Space

After population initialization, the ENAS methods search the architectures in search space. Generally speaking, the search space is the same with initial space when the random initial space is adopted. For other two types of initial space, however, due to the relatively small initial space, the search space will become larger to contain the promising architectures. It's worth noting that many methods do not directly define the search space, but restrict the search space by using evolutionary operators. For example, Irwin *et al.* [141] do not specify the maximum depth, but use the evolutionary operations to extend the architecture to any depth.

IV. ENCODING STRATEGY

This section will discuss how to encode the architecture. Each ENAS method need to determine its encoding strategy before starting the first stage of the ENAS, i.e. the population initialization. The most intuitive difference in different encoding strategies of ENAS methods is the length of the encoded individual.

Commonly, the encoding strategy can be divided into two categories according to whether the length of an individual changes or not during the evolutionary process: *fixed-length encoding strategy* and *variable-length encoding strategy*. The advantage of the *fixed-length encoding strategy* is that it is easy to take the use of standard evolutionary operations. In Genetic CNN [34], for example, the fixed-length binary string helps the evolutionary operators (especially the crossover) become easier to implement. Another example is [114], Loni *et al.* use a fixed-length string of genes to represent the architecture which leading to an easy implementation of the one point crossover on the corresponding encoded information. The advantage of the *variable-length encoding strategy* is that it can contain more details of the architecture with more freedom of expression. However, the evolutionary operators may be not suitable for this kind of encoding strategy and need to be redesigned, where an example can be seen in [20]. Please note that some works claimed their use of the variable-length encoding strategy, where each individual is designed with a maximal length, and then the placeholder is used to indicate the gene's validation of the gene [96]. For example, the individual is designed to have $1,000$ genes, while some genes having the values of zeros do not anticipate the evolutionary process. In this paper, we also categorize these methods which adopt the fixed-length encoding strategy.

In addition, most of the neural architectures can be represented as directed graphs which are made up of different basic units and the connections between the units. Therefore, the encoding for an architecture can be divided into two aspects: configurations of basic units and connections. We will summarize both of them.

A. Encoding for Configurations of Basic Units

In fact, the configurations are not the same in multiple basic units, such as layers, blocks and cells, which are demonstrated in Section III. In CNN, for example, there are multiple parameters in the primitive layers which can be seen in Table I. As for the DenseBlock implemented by Sun *et al.* [9], actually only two parameters are needed to build the block. The configuration of cell is more flexible than block in CNN, it can be regarded as a microcosm of a complete neural network. For example, the cells in [35] are made up by a combination of 10 layers selected from 9 different primitive layers. But the cells do not concern all the configurations of primitive layer, such as the feature map size which is an important parameter in primitive layer [20].

B. Encoding for Connections

When the parameters (configurations) of the basic units in the architecture have been determined, the corresponding architecture can not be built up. Since edges are indispensable in directed graph, connections are also part of the architecture. Actually, the connections discuss in this section including not only connections between basic units, but also connections within the basic units.

Generally, the structure of neural network can be divided into two categories: *linear structure* and *non-linear structure*. The former denotes the architectures containing sequential basic units. The latter indicates that there are skip-connections or loop-connections in the architecture. Please note that, the structure could be the macro structure consisting of basic units, or the micro structure within the basic units.

1) Linear structure: The linear structure can be found in different kinds of architectures including layer-based and block-based. Its widespread use in ENAS stems from its simplicity. No matter how complex the internal of the basic units is, many ENAS methods stack the basic units one by one to build up the skeleton of the architecture which is a linear structure. Such as in AE-CNN [9], Sun *et al.* stack different kinds of blocks to generate the architecture.

One special case is a totally linear architecture, if the layer-based architecture is linear. In this case, there is no need to solely encode the connections, and only the parameters in each basic unit are enough to build an architecture. One classical example can be seen in [20] where Sun *et al.* explore a great many of parameters based on a linear CNN architecture. However, most architectures are not designed to be linear. The skip connection in ResNet [2] and the dense connection in DenseNet [3] show the ability to build a good architecture. In the following, some of the non-linear structure will be listed.

2) Non-linear structure: Firstly, we will introduce two approaches to encode the non-linear structure in this subsection. Adjacent matrix is the most popular way to represent the connections in non-linear structure. Genetic CNN [34] uses a binary string to represent the connection, and the string can transform into a triangular matrix. In the binary string, 1 denotes there is a connection between the two nodes and 0 denotes no connection. Lorenzo *et al.* [134] use a matrix to represent the skip connection, and this work revolves around the adjacent matrix. In fact, back in the 1990s, Kitano *et al.* [142] began to study the use of the adjacent matrix to represent network connection and explained the process from the connectivity matrix, to bit-string genotype, to network architecture phenotype. Another way to represent the connections is using an ordered pair $G = (V, E)$ with vertices V and edge E associated with a direction to represent a directed acyclic graph. Irwin *et al.* [141] use this strategy to encode the connections.

Secondly, since the non-linear structure is a more common case in both macro structure and micro structure. AmoebaNet-A [35], as an example of non-linear macro structure, stack two kinds of cells for several times to build up an architecture. Each cell receives two inputs from the previous two cells separately, which means direct connection and skip-connection are both used in this macro structure. Also in AmoebaNet, each cell is non-linear structure inside, that means the micro structure is also non-linear. The non-linear structure give the architecture more flexibility, which makes it more likely to build up an promising architecture than linear structure.

V. Population Updating

This section is based primarily on the population updating process in Fig. 5. Generally, ENAS algorithms based on different EC paradigms vary greatly in population updating. Table III shows the ENAS algorithms which are classified by the EC method they based and different types of neural networks they searched for. Obviously, the EA-based ENAS algorithms dominate the ENAS. To be more specific, the GA-based ENAS is the most popular by researches which is largely owing to the convenience of architecture representation in GA. Therefore, we will give a detailed introduction to the EA-based ENAS including the selection strategy at first. Later, we will make a brief summery for SI-based ENAS and others separately. The multi-objective ENAS algorithms are also introduced at the end of each subsection.

A. EA for ENAS

The dash box in Fig. 5 shows the general flow of population updating in EA-based ENAS. We will introduce the selection strategy and the evolutionary operations in this subsection. The first stage of population updating is selection. The selection strategies can be classified into several types, where Table IV shows the main kinds of selection strategy. Note that the selection strategy can be not only used in choosing individuals as parents to generate offspring with the evolutionary operators, but also used in environment selection stage which is choosing individuals to make up next population. Zhang *et al.* [103] term these two selections as mate selection and environmental selection separately.

The selection strategy can be divided into five categories: elitism, discard the worst, roulette, tournament selection and others. The simplest strategy is elitism which retains the individuals with higher fitness. Only the best group can survive. However, this can cause a loss of diversity in the population which may lead the population fall into local optima. Discarding the worst is similar to elitism. Real *et al.* [35] using the aging evolution which discards the oldest individual in the population. Aging evolution can explore the search space more, instead of zooming in on good models too early, as non-aging evolution would. The same selection strategy is used in [39]. Zhu *et*

TABLE III
CATEGORIZATION OF EC AND DIFFERENT TYPES OF NEURAL NETWORK

			CNN	DBN	RNN	AE	Others
Single objective	EA	GAs	[9], [19]–[23], [33]–[35], [47], [49], [52], [56], [59]–[61], [63], [65]–[67], [69]–[71], [74], [78], [81], [82], [85], [86], [88], [90], [92], [93], [95], [97]–[99], [103], [107], [110], [113], [117]–[119], [121], [122], [125]–[127], [131], [138], [143], [144]	[39], [145]	[44], [68], [93], [94], [146]	[42]	[83], [139], [147], [148]
		GP	[62], [111], [112], [115], [149], [150]		[151], [152]	[43]	[153]
		ES	[123]		[44], [45], [57], [154]–[156]	[41]	[139], [157]
	SI	ACO	[105]		[135], [137], [158], [159]		
		PSO	[73], [74], [96], [101], [102], [109], [130], [160]	[37], [38]		[24], [46]	[55], [84], [161]
	Other	Memetic	[134], [162]				
		DE	[48], [74]		[89]	[51]	[163]
		HCA	[31], [58], [79]				
		CVOA			[87]		
		Hyper-heuristic		[77]			
		FA	[76]				
		AIS	[75]				
Multi-objective	EA		[64], [72], [80], [100], [104], [106], [114], [114], [120], [128], [133], [136], [164]–[167]	[54], [168]	[169]		[170], [171]
	SI		[91], [108], [132]	[50]			

TABLE IV
SELECTION STRATEGY

Elitism	[31], [41], [42], [45], [52], [58], [66], [75], [79], [86], [111], [115], [132], [134], [153]
Discard the worst or the oldest	[35], [59], [78], [88], [131], [146], [172]
Roulette	[34], [47], [63], [65], [107], [113], [114], [117], [133]
Tournament selection	[9], [19], [20], [22], [23], [62], [82], [94], [99], [118], [121], [126], [128], [129], [172]
Others	[60], [104]

al. [59] combine these two approaches, discarding the worst individual and the oldest individual at the same time. Roulette gives every individual a probability to survive (or be discarded), whether he is the best or not. Tournament selection selects the best one from an equally likely sample of individuals. Furthermore, Johner *et al.* [60] use a ranking function to choose individuals by rank. A selection trick termed as niching is used in [70], [127] to avoid stacking into local optima. This trick allows offspring worse than parent for several generations until evolving to a better one.

Most of the methods focus on preserving the well-performed individuals, however, Liu *et al.* [125] emphasizes the gene more than the survived individuals where the gene can represent any components in the architecture. They believe the individuals which consist of the fine-gene set are more likely to have a promising performance.

In fact, some selection methods aim at preserving the diversity of the population. Elsken *et al.* [104] selects individuals in inverse proportion to their density. Javaheripi *et al.* [173] choose the parents based on the distance (difference) during the mate selection. They choose the two individuals with the highest distance to promote exploration.

In terms of evolutionary operations, mutation and crossover are two of the most common used operations in EA-based ENAS. The mutation is only performed on a single individual, while the crossover takes the use of two individuals to generate offspring.

First, we will discuss the mutation operator. The aim of the mutation operator is to search the local optimum around the individual. A simple idea is to allow the encoded information to vary from a given range. Sun *et al.* [20] use the polynomial mutation [174] on the parameters of layers which are expressed by real numbers. To make the mutation not random, Lorenzo *et al.* [134] proposed a novel Gaussian mutation based on a Gaussian regression to guide the mutation, i.e., the Gaussian regression can predict which architecture may be good, and the new generated individuals are sampled in the regions of the search space where the fitness values are likely to be high. That makes the mutation have a “direction”. Moreover, Maziarz *et al.* [175] use a RNN to guide the mutation. In this work, the mutations are not sampled at random among the possible architectural choices, but are sampled from distributions inferred by an RNN. Actually, using an RNN to control the mutation can be seen in other methods such as [128]. Some researches concern about the diversity of the population after mutation. Qiang *et al.* [38] use a variable mutation probability. They used a higher probability in the early stage for better exploration and a lower probability in the late stage for better exploitation. The fact that it is applied in many other methods [69] is enough to prove its effectiveness. To maintain the diversity of the population after the mutation, Tian *et al.* [66] use force mutation and distance calculation which ensure the individual in the population is not particular similar to other individual (especially the best one). Kramer *et* *al.* [127] use the (1+1)-ES that generates an offspring based on a single parent with bit flip mutation, and use mutation rate control and niching to overcome local optima.

The big computational cost of ENAS is a bottleneck which will be discuss in later sections. In order to reduce the unaffordable computational cost and unbearable time, some kinds of mutation are designed. Zhang *et al.* [103] propose the exchange mutation which exchanges the position of two genes of the individual, i.e. exchange the order of layers. This will not bring new layers and the weight in neural networks can completely be preserved, which means the offspring do not have to be trained from scratch. Chen *et al.* [176] introduced two function-preserving operators on the neural networks, and these operators are termed as network morphisms by Wei *et al.* [177]. The network morphisms aim to change the neural network architecture without loss of the acquired experience. The network morphisms change the architecture from $F(\cdot)$ to $G(\cdot)$, and satisfies the Equation (2):

$$\forall x, \quad F(x) = G(x) \tag{2}$$

where x denotes the input of network. The network morphisms can be regarded as a function-preserving mutation. With this function-preserving mutation, the mutated individual can not have a worse performance than the parent. To be more specific, Chen *et al.* [176] proposed net2widernet to obtain a wider net and net2deepernet to obtain a deeper net. Elsken *et al.* [31] extent the network morphisms with two popular network operations: skip connections and batch normalization. Zhu *et al.* [59] proposed five well-designed function-preserving mutations to guide the evolutionary process by the information which have already learned. To avoid the suboptimal, Chen *et al.* [122] add noises in some function-preserving mutation, and in the experiment they found that by adding noises to pure network morphism, instead of compromising the efficiency, it will, by contrast, improve the final classification accuracy. Noting that all the network morphisms can only increase the capacity of a network, because if one would decrease the network's capacity, the function-preserving property could in general not be guaranteed [104]. And as a result, the architecture generated by network morphisms is only going to get larger and deeper which is not suitable for a device with limited computing resources (like a mobile phone). In order for the network architecture to be reduced, Elsken *et al.* [104] proposed the approximate network morphism, which satisfies the Equation (3):

$$\forall x, \quad F(x) \approx G(x) \tag{3}$$

to also cover operators that reduce the capacity of a neural architecture.

Second, as for the crossover operator, single-point crossover [178] is the most popular method in EA-based ENAS [47], [63], [65], [133] because of its implementation simplicity. However, the single-point crossover can only apply to two individuals with the equal lengths. Therefore, this can not apply to the variable-length individuals. To this end, Sun *et al.* [23] proposed an efficient crossover operator for individuals with unequal lengths. Sapra *et al.* [78] proposed a disruptive crossover swapping the whole cluster (a sequence of layers) between both the individual at corresponding positions rather than only focusing on the parameters of layers. Sun *et al.* [20] using the Simulated Binary Crossover (SBX) [179] to do a combination of the encoded parameters from two matched layers. Please note that, the encoded parameters after SBX are not the same with that of both parents which is quite different from other crossover.

EA for multi-objective ENAS is getting more and more attention from researchers. The single objective ENAS algorithms always concern about one objective function, e.g., the classification accuracy and these algorithms have only one object: searching the architecture with the highest accuracy. Whereas, the multi-objective contains more than one objective functions. In general, most of the multi-objective ENAS algorithms aim at both the performance of neural network and the number of parameters simultaneously [100], [104], [136], [165]. However, these objective functions are often in conflict. For example, getting a higher accuracy often requires a more complicated architecture with the need of more computational resources. On the contrary, a device with limited computational resource, e.g. a mobile phone, can not afford such sophisticated architectures.

The simplest way to tackle the multi-objective optimization is by converted it to the single objective optimization, i.e. the weighting method. The Equation (4)

$$F = \lambda f_1 + (1 - \lambda) f_2 \tag{4}$$

is the classical linear form combining two objective functions f_1, f_2 into a single objective function where the $\lambda \in (0, 1)$ denotes the weights. In [64], [82], [112], [180], the multi-objective optimization problem can be solved by using the available single objective optimization methods by the Equation (4) of the weighted sum. Chen *et al.* [129] do not adopt the linear addition as the objective function, whereas using a nonlinear penalty term. However, the weights manually set may be greatly misleading [181].

Some algorithms have been already widely used in multi-objective optimization, such as NSGA-II [182], MOEA/D [183], which are also used in ENAS methods such as [106], [114], [164]. These methods aim to find a Pareto-font set (or non-dominant set). Only these methods are in the multi-objective category of the Table III. Some researches have made improvements on these multi-objective optimization methods for better use in ENAS. Baldeon *et al.* [106] choose the penalty based boundary intersection approach in MOEA/D because training a neural network involves non-convex optimization and the form of Pareto Font is unknown. LEMONADE [104] divide the objective function into two categories: f_{exp} and f_{cheap}. The f_{exp} denotes expensive-to-evaluate objectives (e.g., the accuracy), the f_{cheap} denotes cheap-to-evaluate objectives (e.g., the model size). In every iteration, they sample parent networks with respect to sparsely distribution based on the cheap objectives f_{cheap} to generate offspring. Therefore, they evaluate the f_{cheap} more times than f_{exp} to save time. Schoron *et al.* [80] also take the use of the LEMONADE proposed by Elsken *et al.* [104]. Due to the NSGA-III [184] may fall into the small model trap (this algorithm prefer the small models), Yang *et al.* [136] have made some improvements to the conventional NSGA-III for protecting these larger models.

B. SI for ENAS

PSO is inspired by the bird flocking or fish schooling [16], and is easy to implement compared with other SI algorithms. Junior *et al.* [73] use their implementation of PSO to update the particle based on the layer instead of the parameters of the layer. Gao *et al.* [160] developed a gradient-priority particle swarm optimization to handle the problem including the low convergence efficiency of PSO when there are a lot of hyper-parameters to be optimized. They expect the particle tends to find the locally optimal solution at first, and then move to the global optimal solution.

For ACO, the individuals are generated in a quite differently way. Several ants are in an ant colony[3]. Each ant move from node to node following the pheromone instructions to build an architecture. The pheromone is updated every generation. The paths of well-performed architecture will maintain more pheromone to attract the next ant for exploitation and at the same time the pheromone is also decaying (i.e. pheromone evaporation) which encourage other ants to explore other areas. Byla *et al.* [105] let the ants choose the path from node to node in a graph whose depth increases gradually. Elsaid *et al.* [158] introduce different ant agent types to act according to specific roles to serve the needs of the colony, which is inspired by the real ants specialize.

SI for multi-objective ENAS started only in the last two years and the research of this field is scarce which can be seen from Table III. Li *et al.* [50] use the bias–variance framework on they proposed multi-objective PSO to get a more accurate and stable architecture. Wu *et al.* [132] use the MOPSO [185] for neural networks pruning. The G_{best} is selected according to the crowding distance in the non-dominant solutions set. Wang *et al.* [108] use the OMOPSO [186] which selects the leaders using a crowding factor and the G_{best} is selected from the leaders. To better control the balance between convergence and diversity, Jiang *et. al.* [91] proposed a MOPSO/D algorithm based on an adaptive penalty-based boundary intersection.

C. Other EC Techniques for ENAS

Different from the GA, the mutation of DE takes the use of the information from three individuals. Some ENAS method like [51], [89] choose DE to guide the offspring generation. However, there is little difference between different DE-based ENAS algorithms.

Moreover, Wang *et al.* [187] proposed a hybrid PSO-GA method. They use PSO to guide the evolution of the parameters in each block encoded in decimal notation, meanwhile use GA to guide the evolution of the shortcut connections encoded in binary notation. Because PSO performs well on continuous optimization and GA is suitable for optimizations with binary values, this hybrid method can search architectures efficiently.

[3]The population in ACO also termed as colony.

There are some other methods not mentioned above. HCA can be interpreted as a very simple evolutionary algorithm. For example, in [31] the evolutionary operators only contains the mutation and no crossover, and the selection strategy is relatively simple. The memetic algorithm is the hybrids of EAs and local search. Evans *et al.* [162] integrate the local search (as gradient descent) into the GP as a fine-tuning operation. The CVOA [87] is inspired by the new respiratory virus, COVID-19. The architecture is found by simulating the virus spreads and infects healthy individuals. Hyper-heuristic contains two levels: high level strategy and low level heuristics, and a domain barrier is between these two levels. Hence the high level strategy is still useful when the applications have changed. AIS is inspired by theories related to the mammal immune system and do not require the crossover operator compared to the GA [75].

VI. Efficient Evaluation

Due to the evaluation is the most time-consuming stage, we will discuss the strategies to improve the efficiency of evaluation in this section.

Real *et al.* [22] use 250 workers (computers) to finish the Large-Scale evolution over 11 days. Such computational resources are not available for anyone who is interested in NAS. Almost all of the methods evaluate individuals by training them first and evaluating them on the validation dataset. Since the architecture is becoming more and more complex, it will take a lot of time training each architecture to convergence. So it is natural to think up the methods to shorten the evaluation time and reduce dependence on large amounts of computational resources. Table V lists five of the most common methods to shorten the time: weight inheritance, early stopping policy, reduced training set, reduced population and population memory. We would like to introduce the five kinds of methods first, other kinds of promising methods are introduced next, especially the surrogate-assisted methods at the end of this section.

Because the evolutionary operators usually do not completely disrupt the architecture of an individual, some parts of the new generated individual are the same with previous individuals. The weights of the same parts can be easily inherited. With the weight inheritance, the neural networks are no longer trained completely from scratch. This method has been used in [161] 20 years ago. Moreover, as mentioned in Section V, the network morphisms change the network architecture without loss of the acquired experience. This could be regarded as the ultimate weight inheritance, because it solved the weight inheritance problem in the changed architecture part. The ultimate weight inheritance let the new individual completely inherit the knowledge its parent learned and training such an individual to convergence will save a lot of time.

Early stopping policy is another method which is used widely in NAS. The simplest way is to set a fixed relatively small number of training epochs. This method is used in [24], because the authors think the individual after a small number of training epochs can conduct the performance. Similarly, Assunccao *et al.* [81] let the individuals undergo the training for a same and short time each epoch (although this time is not fixed and will increase with the epoch). To let the promising architecture have more training time to get a more precise evaluation, So *et al.* [188] set hurdles after several fixed epochs. The weak individuals stop training early and save the time. However, the early stopping policy can lead to inaccurate estimation about individual performance (especially the large and complicated architecture), which can be seen in Fig. 7. In Fig. 7, *individual2* performs better than *individual1* before epoch *t1*, whereas *individual1* performs better in the end. Yang *et al.* also discuss this phenomenon in [136]. So, it is crucial to determine at which point to stop. Note that the neural network can converge or hardly improve its performance after several epochs, such as the *t1* for *individual2* and the *t2* for *individual1* in Fig. 7. Using the performance estimated at this point can evaluate an individual relatively accurately with less training time. Therefore, some methods such as [33], [88] stop training when observing there is no significant performance improvement. Suganuma *et al.* [115] use the early stopping policy based on a reference curve. If the accuracy curve of an individual is under the reference curve for successive epochs, then the training will be terminated and this individual is regarded as poor one. After every epoch, the reference curve is updated by the accuracy curve of the best offspring.

Reduced training set, i.e. using a subset of that data has similar properties to a large dataset can also shorten the time effectively. Liu *et al.* [125] explore the promising architecture by training on a subset and use the transfer learning to the large original dataset. Because there are so many benchmark datasets in image classification field, the architecture can be evaluated on the smaller dataset (e.g. CIFAR-10) first and then is applied on the large dataset (such as CIFAR-100 and ImageNet). The smaller dataset can be regarded as the proxy for the large one.

Reduced population is the unique method of ENAS, since other NAS do not have the population. Assunccao *et al.* [98] reduce the population on the basis of their previous algorithm [189], [190] to speed up the evolution. However, simply reduced population may not explore the search space dramatically in each epoch and may lose the global search ability. Another way is reducing the population dynamically. For instance, Fan *et al.* [123] use the $(\mu + \lambda)$ evolution strategy and divide the evolution into three stages with the population reduction which aim to find the balance of the limited computing resources and the efficiency of evolution. The large population in the first stage is to ensure the global search ability, while the small population in the last stage is to shorten the evolution time. Instead of reducing the population, Liu *et al.* [191] evaluate the architecture with small size at an early stage of evolution. Similarly, Wang *et al.* [130] do not evaluate the whole architecture whereas a single block, and then the blocks are stacked to build an architecture as evolution goes on.

The population memory is another category of unique method of ENAS. It works by reusing the corresponding architectural information that has previously appeared in the population. In population based methods, especially the GA based methods (e.g., in [20]), it is natural to maintain well performed individuals in the population in successive epochs. Sometimes, the individuals in the next population

TABLE V
DIFFERENT METHODS TO SHORTEN THE EVALUATION TIME

Weight inheritance	[22], [31], [58], [59], [65], [70], [75], [79], [80], [86], [88], [90], [104], [105], [109], [121], [122], [126], [132], [134]
Early stopping policy	[24], [33], [46], [65], [66], [68], [74], [75], [81], [82], [85], [88], [88], [96], [98], [102], [108], [112], [115], [126], [160]
Reduced training set	[49], [86], [125], [130], [191]
Reduced population	[98], [123], [125]
Population memory	[23], [33], [60], [118]

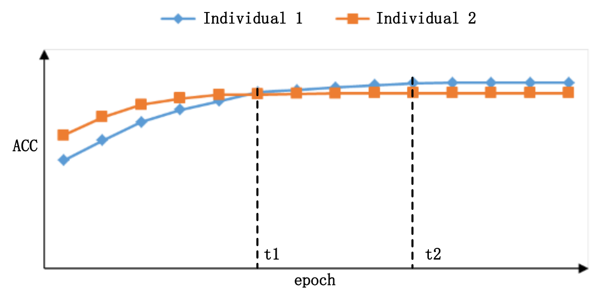

Fig. 7. Two learning curve of two different individuals.

directly inherit all the architecture information of their parents without any modification and it is not necessary evaluating the individuals again. Fujino *et al.* [33] use a memory to record the fitness of individuals, and if the same architecture encoded in individual appears, the fitness value is retrieved from memory instead of being reevaluated. Similarly, Miahi *et al.* [118] and Sun *et al.* [23], employ a hashing method for saving pairs of architecture and fitness of each individual and reusing them when the same architecture appears again. Johner *et al.* [60] prohibit the appearance of architectures that have existed before in offspring. This does reduce the time, however, the best individuals are prohibited from remaining in the population which may lead the population evolving towards a bad direction.

Actually, there are many other well performed methods to reduce the time in ENAS. Rather than training thousands of different architectures, one-shot model [192] trains only one SuperNet to save the time. The different architectures, i.e. the SubNets, are sampled from the SuperNet with the shared parameters. Yang *et al.* [136] believe the traditional ENAS methods without using SuperNet are less efficient for models are optimized separately. In contrast, the one-shot model optimize the architecture and the weights alternatively. But the weight sharing mechanism brings a difficulty in accurately evaluating the architecture. Chu *et al.* [193] scrutinize the weight-sharing NAS with a fairness perspective and demonstrate the effectiveness. However, there remains some doubts that can not explain clearly in one-shot model. The weights in the SuperNet are coupled. It is unclear why inherited weights for a specific architecture are still effective [138].

Taking the use of hardware can reduce the time, too. Jiang *et al.* [91] use a distributed asynchronous system which contains a major computing node with 20 individual workers. Each worker is responsible for training a single block and uploading its result to the major node in every generation. Wang *et al.* [108] design an infrastructure which has the ability to leverage all of the available GPU cards across multiple machines to concurrently perform the objective evaluation for a batch of individuals. Note that Colangelo *et al.* [194], [195] design a reconfigurable hardware framework that fits the ENAS. As they claimed, this is the first work of conducting NAS and hardware co-optimization.

Furthermore, Lu *et al.* [100] adopt the concept of a proxy models which are small-scale versions of the intended architectures. For example, in a CNN architecture, the number of layers and the number of channels in each layer are reduced. However, the drawback of this method is obvious: the loss of prediction accuracy. Therefore, they perform an experiment to determine the smallest proxy model that can provide a reliable estimate of performance at a larger scale.

All the above methods obtain the fitness of individuals by directly evaluating the performance on the validation dataset. An alternative way is using the indirect methods, namely the performance predictors. As summarized in [196], the performance predictors can be divided into two categories: performance predictors based on the learning curve and end-to-end performance predictors, both of which are based on the training-predicting computational paradigm. This does not mean the performance predictor does not undergo the training phase at all, while it means learning from the information obtained in the training phase, and use the knowledge learned to make a reasonable prediction for other architectures. Ralwal *et al.* [152] take the use of the learning curve based predictor where the fitness is not calculated in the last epoch but is predicted by the sequence of fitness from first epochs. Specifically, they use a Long Short Term Memory (LSTM) [197] as a sequence to sequence model, when input the validation perplexity of the first epochs, the output will be the validation perplexity prediction of the subsequent epochs. Sun *et al.* [196] use an end-to-end performance predictor which does not need any extra information, even the performance of individuals of the first several epochs. Specifically, they adopt a method based on the random forest to accelerate the fitness evaluation in ENAS. When the random forest receives a newly generated architecture as input, adaptive combination of a huge number of regression trees which have been trained in advance in the forest gives the prediction.

VII. Applications

This section lists different fields ENAS involved. Actually, the ENAS can be applied to wherever the neural networks can be applied. The following Table VI shows the wide range of applications and Table VII displays the performance of extraordinary ENAS methods on two popular and challenging dataset for image classification, namely CIFAR-10 and CIFAR-100. Both of these two tables can represent what ENAS has achieved so far.

A. Overview

Table VI shows the applications of ENAS. This is a incomplete statistics but also contains a wide range of applications.

TABLE VI
Application

Category	Applications	References
1	Image classification	[9], [20]–[24], [33]–[35], [42], [43], [46], [47], [52], [53], [59]–[61], [64]–[66], [70], [71], [73], [74], [74]–[76], [80], [82], [85], [86], [90]–[92], [96]–[101], [103]–[105], [107]–[112], [114]–[116], [119], [121], [122], [124], [126], [127], [129]–[134], [136], [138], [141], [149], [157], [162], [164], [165], [167], [173], [175], [195], [198]–[201]
1	Image to image	[41], [49], [95], [113], [120], [128], [144]
1	Emotion recognition	[93], [160]
1	Speech recognition	[57], [83], [102], [139], [155]
1	Language modeling	[116], [152]
1	Face De-identification	[202]
2	Medical image segmentation	[38], [39], [72], [106], [117], [123], [134]
2	Malignant melanoma detection	[58], [79]
2	Sleep heart study	[163]
2	Assessment of human sperm	[118]
3	Wind speed prediction	[154]
3	Electricity demand time series forecasting	[87]
3	Traffic flow forecasting	[50]
3	Electricity price forecasting	[89]
3	Car park occupancy prediction	[44]
3	Energy consumption prediction	[94]
3	Time series data prediction	[146]
3	Financial prediction	[203]
3	Usable life prediction	[54]
3	Municipal waste forecasting	[45]
4	Engine vibration prediction	[135], [159]
4	UAV	[148]
4	Bearing fault diagnosis	[51]
4	Predicting general aviation flight data	[137]
5	Crack detection of concrete	[63]
5	Gamma-ray detection	[81]
5	Multitask learning	[204]
5	Identify Galaxies	[143]
5	Video understanding	[69]
5	Comics understanding	[56], [205]

Generally, these fields of application can be grouped into the following five categories:

(1) Image and signal processing, including image classification which is the most popular and competitive field, image to image processing (including image restoration, image denoising, super-resolution and image inpainting), emotion recognition, speech recognition, language modeling and face de-identification.

(2) Biological and biomedical tasks, including medical image segmentation, malignant melanoma detection, sleep heart study and assessment of human sperm.

(3) Predictions and forecasting about all sorts of things, including the prediction of wind speed, car park occupancy, time series data, financial and usable life, the forecasting of electricity demand time series, traffic flow, electricity price and municipal waste.

(4) Machine, including engine vibration prediction, Unmanned Aerial Vehicle (UAV), bearing fault diagnosis and predicting general aviation flight data.

(5) Others, including crack detection of concrete, gamma-ray detection, multitask learning, identify galaxies, video understanding and comics understanding.

B. Comparison on CIFAR-10 and CIFAR-100

In Table VI, it is obviously to see that many ENAS methods are applied on the image classification task. The benchmark dataset, CIFAR-10 which contains a total of ten classes, and the CIFAR-100 is the advanced dataset including a hundred of classes. These two datasets are widely used in image classification tasks, and the accuracy on these two challenging datasets can represent the ability of the architecture. We collect the well-performed ENAS methods tested on these two datasets. The Table VII shows the test results on the two datasets of different state-of-the-art methods separately, where the methods are ranked in ascending order of their best accuracy on CIFAR-10, i.e. the methods are ranked in descending order of their error rate. The column "CIFAR-10" and "CIFAR-100" denote the error rate of each method on the corresponding datasets respectively. Furthermore, the "GPU Days" denotes the total search time of each method, it can be calculated by the Equation (5)

$$GPU\ Days = The\ number\ of\ GPUs \times t \tag{5}$$

where the t denotes the search time of the method. "Parameters" denotes the total number of parameters which can represent the capability of an architecture and the complexity. In addition, the symbol "—" in Table VII implies there is no result publicity reported by the corresponding literature. The year reported in this Table is its earliest public time.

Actually, this is not a totally fair comparison. The reason can be summarized in the following two aspects: (1) The encoding space including the initial space and the search space are not the same. There are two extreme cases in initial space: trivial initialization which starts at the simplest architecture and rich initialization which starts at a well-designed architecture (e.g. ResNet-50 [2]). And the size of search space is largely different, e.g., Ref [47] only takes the kernel size into search space. (2) Different tricks used in the methods, e.g. the "cutout", can make the final results unfair, too. The "cutout" refers to a regularization method [208] used in the training of CNNs, which could improve the final performance.

Anyhow, Table VII shows the progress of ENAS in image classification according to the accuracy on CIFAR-10: Large-Scale Evolution [22] (5.4%, 2017), LEMONADE [104] (2.58% 2018), NSGANet [100] (2.02%, 2019). Many ENAS methods have lower error rate than ResNet-110 [2] with 6.43% error rate on CIFAR-10, which is a manually well-designed architecture. Therefore, the architecture found by ENAS can reach the same level as or exceed the architecture designed by experts. It proves that the ENAS is reliable and can be used in other application field.

VIII. Challenges and Issues

Despite the positive results of the existing methods, there are still some challenges and issues which need to be addressed.

A. The Effectiveness

The effectiveness of ENAS is questioned by many researches. Wistuba *et al.* [26] notice that the random search can get a well-performed architecture and has proven to be an extremely strong baseline. Yu *et al.* [209] show the state-of-the-art NAS algorithms perform similarly to the random policy on average. Liashchynskyi *et al.* [210] compare grid search, random search, and GA for NAS and the result is that the architecture obtained by GA and the random search have similar performance. There is no need to use the complicate algorithms to guide the search process if the random search can outperform the NAS based on EC paradigm.

However, the evolutionary operator in [210] only contains a recombination operator which could not represent the whole effectiveness of ENAS. Although the random search can find a well-performed architecture in experiment, it can not guarantee that it will find a good architecture every time. Moreover, in [99], [138], the evolutionary search is more effective than random search. There is an urgent need to design a sophisticated experiment to tell the effectiveness of the state-of-the-art ENAS methods, especially in a large encoding space.

B. The Mutation and the Crossover

In Section V, two types of operators are introduced. We note that some methods like Large-Scale NAS [22] only use single individual based operator (mutation) to generate offspring. The main reason why they do not choose the crossover operator in their method come from two aspects: the first is for simplicity [169], and the second is that simply combining a section of one individual with a section of another individual seems ill-suited to the neural network paradigm [75]. Also, in [26], the authors believe that there is no indication that a recombination operation applied to two individuals with high fitness would result into an offspring with similar or better fitness.

However, the supplemental materials in [23] demonstrate the effectiveness of the crossover operator in this method. This method can find a good architecture in short order with the help of the crossover. On the contrary, when the crossover is not performed, the architecture found is not really promising, unless it runs for a long time. In fact, the mutation operator let an individual explore the space around itself, and it is a gradually incremental search process like searching step by step. The crossover (recombination) can generate offspring dramatically different from the parents by contrast, which is more like a stride. So, this operator has the ability to efficiently find a promising architecture. Chu *et al.* [120] prefer that while a crossover mainly contributes to exploitation, a mutation is usually aimed to introduce exploration. These two operators play different role in the evolutionary process. But there is not a sufficient explanation how the crossover operator works. Maybe some additional experiments need to be done on the methods that do not include the crossover operator.

TABLE VII
THE COMPARISON OF THE ERROR RATE ON CIFAR-10 AND CIFAR-100

ENAS Methods	GPU Days	Parameters (M)	CIFAR-10(%)	CIFAR-100 (%)	Year	Notes
CGP-DCNN [112]	—	1.1	8.1	—	2018	We choose the architecture with the lowest error rate.
EPT [86]	2	—	7.5	—	2020	
GeNet [34]	17	—	7.1	—	2017	The report starting from the 40-layer wide residual network is unadopted.
		—	—	29.03		
EANN-Net [107]	—	—	7.05 ± 0.02	—	2019	
DeepMaker [114]	3.125	1	6.9	—	2020	We choose the two architectures with the lowest error rate.
		1.89	—	24.87		
GeneCai (ResNet-50) [173]	0.024	—	6.4	—	2020	The cost of this model is 30% of the origial ResNet-50. (The cost is based on the non-zero parameter ratio/FLOPs)
CGP-CNN (ConvSet) [111]	—	1.52	6.75	—	2017	
CGP-CNN (ResSet) [111]	—	1.68	5.98	—	2017	
MOPSO/D-Net [91]	0.33	8.1	5.88	—	2019	
ReseNet-50 (20% pruned) [133]	—	6.44	5.85	—	2018	The number of parameters decreased by 74.9%.
ImmuNeCS [75]	14	—	5.58	—	2019	
EIGEN [92]	2	2.6	5.4	—	2018	
	5	11.8	—	21.9		
Large-Scale Evolution [22]	2750	5.4	5.4	—	2017	
		40.4	—	23		
CGP-CNN (ConvSet) [115]	31	1.5	5.92 (6.48 ± 0.48)	—	2019	We report the classification errors in the format of "best (mean ± std)." And this is different from the previous CGP-CNN.
	—	2.01	—	26.7 (28.1 ± 0.83)		
CGP-CNN (ResSet) [115]	30	2.01	5.01 (6.10 ± 0.89)	—		
	—	4.6	—	25.1 (26.8 ± 1.21)		
CNN-GA [23]	35	2.9	4.78	—	2018	We do not choose the architecture with cutout.
	40	4.1	—	22.03		
MOCNN [108]	24	—	4.49	—	2019	
NASH [31]	4	88	4.4	—	2017	
	5	111.5	—	19.6		
HGAPSO [74]	7+	—	4.37	—	2018	7+ denotes the GPU Days is more than 7.
DPP-Net [206], [207]	2	11.39	4.36	—	2018	We choose the most representative two architectures.
		0.45	5.84	—		
AE-CNN [9]	27	2	4.3	—	2019	
	36	5.4	—	20.85		
SI-ENAS [103]	1.8	—	4.07	18.64	2020	
EPSOCNN [130]	4-	6.77	3.69	—	2019	4- denotes the GPU Days is less than 4.
NSGA-Net [165]	8	3.3	3.85	20.74	2018	We do not report the results using cutout. And the architecture performs on CIFAR-100 is transferred from that on CIFAR-10.
Hierarchical Evolution [99]	300	—	3.63 ± 0.10	—	2017	
EA-FPNN [126]	0.5	5.8	3.57	—	2018	We report the best run of the method.
	1	7.2	—	21.74		
AmoebaNet-A [35]	3150	3.2	3.34 ± 0.06	—	2018	We report the architecture with lower error rate.
Firefly-CNN [76]	—	3.21	3.3	22.3	2019	It has not been named yet and we name it because of the firefly algorithm.
JASQNet [129]	3	3.3	2.9	—	2018	This method train with cutout.
	3	1.8	2.97	—		
RENASNet [121]	6	3.5	2.88 ± 0.02	—	2018	
CARS [136]	0.4	2.4	3	—	2019	We choose the most representative two architectures.
		3.6	2.62	—		
LEMONADE [104]	80	13.1	2.58	—	2018	We choose the most representative two architectures.
		0.5	4.57	—		
EENA [59]	0.65	8.47	2.56	—	2019	The architecture performs on CIFAR-100 is transferred from that on CIFAR-10.
		8.49	—	17.71		
EEDNAS-NNMM [122]	0.5	4.7	2.55	—	2019	We name it after the first letter of the paper.
NSGANet [100]	27	0.2	4.67	—	2019	We choose the one with the lowest error rate and the one with the least number of parameters on CIFAR-10 and CIFAR-100 seperately.
		4	2.02	—		
		0.2	—	25.17		
		4.1	—	14.38		

C. Scalability

The scale of the datasets used in most ENAS methods is relatively large. Taking the image classification task as an example, the MNIST dataset [211] is one of the earliest datasets. In all, it contains $70,000$ 28×28 grayscale images. Later in 2009, CIFAR-10 and CIFAR-100 [13] including $60,000$ 32×32 color images are medium-scale datasets. One of the most well-known large-scale datasets is the ImageNet [212], which provides more than 14 million manually annotated high-resolution images. Unlike CIFAR-10 and CIFAR-100, which are commonly used in ENAS, fewer methods choose to verify their performance on ImageNet [121], [133], [175]. This can be explained by Table VII, where the GPU Days are usually tens or even thousands. It will be unaffordable for majority of researchers when the medium-scale dataset changes to the lager one.

However, Chen *et al.* [121] believed that the results on the larger datasets like ImageNet are more convincing, because CIFAR-10 is easy to be over-fitting. A popular way to deal this is using a proxy on CIFAR-10 and transfer to ImageNet [35], [104]. Another alternative approach is to use the down-scaled dataset such as ImageNet64×64 [213].

D. Efficient Evaluation Method and Reduce Computational Cost

Section VI has introduced the most popular and effective ways to reduce the evaluation time and the computational cost. In a nutshell, it can be described as a question that strikes the balance between the time spent and the accuracy of the evaluation. Because of the unbearable full training time, we must compromise as little as we can on evaluation accuracy in exchange for significantly reduction in evaluation time without sufficient computing resources.

Although a lot of ENAS methods have adopted various kinds of ways to shorten the evaluation time and even though Sun *et al.* [196] specifically proposed a method of acceleration, the research direction of search acceleration is just getting started. The current approaches have many limitations that need to be addressed. For example, although the Large-Scale Evolution [22] use the weight inheritance to shorten the evaluation time and reduce computational cost, it still runs for several days with the use of lots of computational resources which can not easily access by general researches. Furthermore, there is no baseline and common assessment criteria of the search acceleration methods. It is a big challenge to propose a novel kind of method to evaluate the architecture accurately and quickly.

E. Interpretability

CNN is worked known with its black-box-like solutions, which is hard to interpret [62]. Although some works have done to visualize the process of feature extraction [214], it becomes still uninterpretable due to the large number of learned features [149]. The low interpretability of the manual designed architecture becomes a big obstacle to the development of neural networks. To overcome this obstacle, some researches [62], [149], [162] used GP to automatically design the neural network. Well-known with the high interpretability, GP aims at solving problems by automatically evolving computer programs [215].

The above researches [62], [149], [162] all give a further analysis to demonstrate the high interpretability. Specifically, Evans *et al.* [149] have made a visualization on the JAFFE dataset [216] to expound how the evolved convolution filter served as a form of edge detection, and the large presence of white in output of the convolution can help the classification. In their subsequent work [162], they make a visualization of the automatically evolved model on the Hands dataset [217], where the aggregation function extracts the minimum value of a specific area in the hand image to determine whether the hand is open or closed. Furthermore, Bi *et al.* [62] display the features described by the evolved functions like convolution, max-pooling and addition, and the generated salient features are discriminative for face classification.

Despite the high interpretability the existing work made, all these GP-based ENAS methods only aim at shallow CNN and the number of the generated features is relatively small. However, the most successful CNNs all have a deep architecture. It is necessary to use deep-GP to evolve a deeper GP tree and make a further analysis on the deeper architecture in the future.

F. Future Application

Table VI shows variable applications which are researched by the ENAS currently. But these are just a small part of all areas of neural network application. Actually, ENAS can be applied wherever neural networks can be applied and automate the process of architecture designed which should have done by experts. Moreover, plenty of the image classification successes of ENAS have proven the ENAS has the ability to replace experts. The automated architecture design is a trend.

However, this process is not totally automated. The encoding space (search space) still needs to be designed by experts for different applications. For example, for the image processing tasks, the CNNs are more suitable, so the encoding space contains the layers including convolution layers, pooling layers and fully connected layers. While for the time series data processing, the RNNs are more suitable, so the encoding space may contain the cells including Δ-RNN cell, LSTM [197], Gated Recurrent Unit (GRU) [218], Minimally-Gated Unit (MGU) [219] and Update-Gated RNN (UGRNN) [220]. The two manually determined encoding space already contains a great deal of artificial experience and the components without guaranteed performance are excluded. The problem is: can a method search the corresponding type of neural network for multiple tasks in a large encoding space including all the popular wide used components? Instead of searching one multitask networks [204] which learns several tasks at once with the same neural network, the aim is to find appropriate networks for different tasks in one large encoding space.

G. Fair Comparisons

Section VII-B gives a brief introduction of the unfair comparison. The unfairness mainly comes from two aspects: (1) the tricks including cutout [208], ScheduledDropPath [221], etc. (2) The different encoding space. For aspect (1), we notice some ENAS methods [23] have reported the results with and without the tricks. As aspect (2), the well-designed search space is widely used in different ENAS methods. For instance, the NASNet search space [221] is also used in [35], [131] because it is well-constructed even that random search can perform well. The comparison under the same condition can tell the effectiveness of different search methods.

Fortunately, the first public benchmark dataset for NAS, the NAS-Bench-101 [222] has been proposed. The dataset contains 432k unique convolutional architectures based on the cell-based encoding space. Each architecture can query the corresponding metrics, including test accuracy, training time, etc., directly in the dataset without the large-scale computation. NAS-Bench-201 [223] is proposed recently and is based on another cell-based encoding space which has no limitations on edges. Compared with NAS-Bench-101, which was only tested on CIFAR-10, this dataset collects the test accuracy on three different image classification datasets (CIFAR-10, CIFAR-100, ImageNet). But the encoding space is relatively small, and only contains 15.6k architectures. Actually, experiment with different ENAS methods on these benchmark datasets can get a fair comparison and it will not take too much time. However, these datasets are only based on the cell-based encoding space and can not contain all the search space of the existing methods, because the other basic units (layers and blocks) are built by more hyper-parameters which may lead to a larger encoding space.

In the future, a common platform for the comparison needs to be built. This platform must have several benchmarks encoding space, such as the NASNet search space, NAS-Bench-101 and NAS-Bench-201. All the ENAS methods can directly test on the platform. Furthermore, this platform also needs to solve the problem that the different kinds of GPUs have different computing power which may cause an inaccurate GPU Days based on the different standards. The GPU Days can not compare directly until they have a common base line of computing power.

IX. Conclusions

This paper provided a comprehensive survey of ENAS. We introduced the ENAS from four aspects: population representation, encoding space, population updating and efficient evaluation following the unified flow which can be seen in Fig. 5. The various applications and the performance of the state-of-the-art methods on image classification are also summarized in tables to demonstrate the wide applicability and the promising ability. Challenges and issues are discussed to identify the future research in this field.

To be specific, firstly, encoding space is introduced by categories. We divide encoding space into two parts: initial space and search space where the former one defines the initial conditions whereas the latter one limits the architectures can be found in the evolutionary process. In addition, different encoding strategies and architecture representations are introduced. Secondly, the process of population updating including the evolutionary operators, the multi-objective search strategy and the selection strategy are presented. A variety of EC paradigms use respective regulation to generate new individuals. Based on the original algorithms, many novel methods have proposed improvement to get a stable and reliable search capability. Furthermore, we demonstrated the existing methods to relax the demanding requirement of the long evaluation time and the computational cost which is a huge obstacle to efficiency.

Despite the state-of-the-art methods have achieved some success, the ENAS still faces challenges and issues. The first important issue is whether the EC-based search strategy is useful. If the result is at the same level of baseline (e.g., random search), it is unnecessary to design the complex evolutionary operators. A sophisticated experiment is in urgent need to tell the effectiveness, especially in a large encoding space. Secondly, the crossover operator is an undirected multiple individual based operator and there is no sufficient explanation how the crossover operator works. Besides, the ENAS is just beginning a new era, so that there is a lot of uncharted territory to be explored. Moreover, a unified standard or platform is demanded to make a fair comparison.

References

[1] Y. LeCun, Y. Bengio, and G. Hinton, “Deep learning,” *nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[2] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[3] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, “Densely connected convolutional networks,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 4700–4708.

[4] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, “Bert: Pre-training of deep bidirectional transformers for language understanding,” *arXiv preprint arXiv:1810.04805*, 2018.

[5] Y. Zhang, W. Chan, and N. Jaitly, “Very deep convolutional networks for end-to-end speech recognition,” in *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2017, pp. 4845–4849.

[6] Y. Bengio, “Practical recommendations for gradient-based training of deep architectures,” in *Neural networks: Tricks of the trade*. Springer, 2012, pp. 437–478.

[7] J. K. Kearney, W. B. Thompson, and D. L. Boley, “Optical flow estimation: An error analysis of gradient-based methods with local optimization,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, no. 2, pp. 229–244, 1987.

[8] K. Simonyan and A. Zisserman, “Very deep convolutional networks for large-scale image recognition,” *arXiv preprint arXiv:1409.1556*, 2014.

[9] Y. Sun, B. Xue, M. Zhang, and G. G. Yen, “Completely automated cnn architecture design based on blocks,” *IEEE transactions on neural networks and learning systems*, 2019.

[10] B. Zoph and Q. V. Le, “Neural architecture search with reinforcement learning,” *arXiv preprint arXiv:1611.01578*, 2016.

[11] L. P. Kaelbling, M. L. Littman, and A. W. Moore, “Reinforcement learning: A survey,” *Journal of artificial intelligence research*, vol. 4, pp. 237–285, 1996.

[12] T. Bäck, D. B. Fogel, and Z. Michalewicz, “Handbook of evolutionary computation,” *Release*, vol. 97, no. 1, p. B1, 1997.

[13] A. Krizhevsky, G. Hinton *et al.*, “Learning multiple layers of features from tiny images,” 2009.

[14] D. E. Goldberg, *Genetic algorithms*. Pearson Education India, 2006.

[15] W. Banzhaf, P. Nordin, R. E. Keller, and F. D. Francone, *Genetic programming*. Springer, 1998.

[16] J. Kennedy and R. Eberhart, “Particle swarm optimization,” in *Proceedings of ICNN’95-International Conference on Neural Networks*, vol. 4. IEEE, 1995, pp. 1942–1948.

[17] Y. Sun, G. G. Yen, and Z. Yi, “Igd indicator-based evolutionary algorithm for many-objective optimization problems,” *IEEE Transactions on Evolutionary Computation*, vol. 23, no. 2, pp. 173–187, 2018.

[18] A. Darwish, A. E. Hassanien, and S. Das, “A survey of swarm and evolutionary computing approaches for deep learning,” *Artificial Intelligence Review*, vol. 53, no. 3, pp. 1767–1812, 2020.

[19] E. Dufourq and B. A. Bassett, “Automated problem identification: Regression vs classification via evolutionary deep networks,” in *Proceedings of the South African Institute of Computer Scientists and Information Technologists*, 2017, pp. 1–9.

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

[21] A. Martín, R. Lara-Cabrera, F. Fuentes-Hurtado, V. Naranjo, and D. Camacho, “Evodeep: a new evolutionary approach for automatic deep neural networks parametrisation,” *Journal of Parallel and Distributed Computing*, vol. 117, pp. 180–191, 2018.

[22] E. Real, S. Moore, A. Selle, S. Saxena, Y. L. Suematsu, J. Tan, Q. V. Le, and A. Kurakin, “Large-scale evolution of image classifiers,” in *Proceedings of the 34th International Conference on Machine Learning-Volume 70*. JMLR. org, 2017, pp. 2902–2911.

[23] Y. Sun, B. Xue, M. Zhang, G. G. Yen, and J. Lv, “Automatically designing cnn architectures using the genetic algorithm for image classification,” *IEEE Transactions on Cybernetics*, 2020.

[24] Y. Sun, B. Xue, M. Zhang, and G. G. Yen, “A particle swarm optimization-based flexible convolutional autoencoder for image classification,” *IEEE transactions on neural networks and learning systems*, vol. 30, no. 8, pp. 2295–2309, 2018.

[25] T. Elsken, J. H. Metzen, and F. Hutter, “Neural architecture search: A survey,” *arXiv preprint arXiv:1808.05377*, 2018.

[26] M. Wistuba, A. Rawat, and T. Pedapati, “A survey on neural architecture search,” *arXiv preprint arXiv:1905.01392*, 2019.

[27] K. O. Stanley, J. Clune, J. Lehman, and R. Miikkulainen, “Designing neural networks through neuroevolution,” *Nature Machine Intelligence*, vol. 1, no. 1, pp. 24–35, 2019.

[28] M. Dorigo, V. Maniezzo, and A. Colorni, “Ant system: optimization by a colony of cooperating agents,” *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 26, no. 1, pp. 29–41, 1996.

[29] K. Price, R. M. Storn, and J. A. Lampinen, *Differential evolution: a practical approach to global optimization*. Springer Science & Business Media, 2006.

[30] X.-S. Yang, “Firefly algorithms for multimodal optimization,” in *International symposium on stochastic algorithms*. Springer, 2009, pp. 169–178.

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

[32] H. Wang, D. Wang, and S. Yang, “A memetic algorithm with adaptive hill climbing strategy for dynamic optimization problems,” *Soft Computing*, vol. 13, no. 8-9, pp. 763–780, 2009.

[33] S. Fujino, N. Mori, and K. Matsumoto, “Deep convolutional networks for human sketches by means of the evolutionary deep learning,” in *2017 Joint 17th World Congress of International Fuzzy Systems Association and 9th International Conference on Soft Computing and Intelligent Systems (IFSA-SCIS)*. IEEE, 2017, pp. 1–5.

[34] L. Xie and A. Yuille, “Genetic cnn,” in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 1379–1388.

[35] E. Real, A. Aggarwal, Y. Huang, and Q. V. Le, “Regularized evolution for image classifier architecture search,” in *Proceedings of the aaai conference on artificial intelligence*, vol. 33, 2019, pp. 4780–4789.

[36] G. E. Hinton, S. Osindero, and Y.-W. Teh, “A fast learning algorithm for deep belief nets,” *Neural computation*, vol. 18, no. 7, pp. 1527–1554, 2006.

[37] J. K. Kim, Y. S. Han, and J. S. Lee, “Particle swarm optimization–deep belief network–based rare class prediction model for highly class imbalance problem,” *Concurrency and Computation: Practice and Experience*, vol. 29, no. 11, p. e4128, 2017.

[38] N. Qiang, B. Ge, Q. Dong, F. Ge, and T. Liu, “Neural architecture search for optimizing deep belief network models of fmri data,” in *International Workshop on Multiscale Multimodal Medical Imaging*. Springer, 2019, pp. 26–34.

[39] W. Zhang, L. Zhao, Q. Li, S. Zhao, Q. Dong, X. Jiang, T. Zhang, and T. Liu, “Identify hierarchical structures from task-based fmri data via hybrid spatiotemporal neural architecture search net,” in *International Conference on Medical Image Computing and Computer-Assisted Intervention*. Springer, 2019, pp. 745–753.

[40] I. Goodfellow, Y. Bengio, A. Courville, and Y. Bengio, *Deep learning*. MIT press Cambridge, 2016, vol. 1.

[41] M. Suganuma, M. Ozay, and T. Okatani, “Exploiting the potential of standard convolutional autoencoders for image restoration by evolutionary search,” *arXiv preprint arXiv:1803.00370*, 2018.

[42] J. Hajewski and S. Oliveira, “An evolutionary approach to variational autoencoders,” in *2020 10th Annual Computing and Communication Workshop and Conference (CCWC)*. IEEE, 2020, pp. 0071–0077.

[43] L. Rodriguez-Coayahuitl, A. Morales-Reyes, and H. J. Escalante, “Evolving autoencoding structures through genetic programming,” *Genetic Programming and Evolvable Machines*, vol. 20, no. 3, pp. 413–440, 2019.

[44] A. Camero, J. Toutouh, D. H. Stolfi, and E. Alba, “Evolutionary deep learning for car park occupancy prediction in smart cities,” in *International Conference on Learning and Intelligent Optimization*. Springer, 2018, pp. 386–401.

[45] A. Camero, J. Toutouh, and E. Alba, “A specialized evolutionary strategy using mean absolute error random sampling to design recurrent neural networks,” *arXiv preprint arXiv:1909.02425*, 2019.

[46] Y. Sun, B. Xue, M. Zhang, and G. G. Yen, “An experimental study on hyper-parameter optimization for stacked auto-encoders,” in *2018 IEEE Congress on Evolutionary Computation (CEC)*. IEEE, 2018, pp. 1–8.

[47] A. Singh, S. Saha, R. Sarkhel, M. Kundu, M. Nasipuri, and N. Das, “A genetic algorithm based kernel-size selection approach for a multi-column convolutional neural network,” *arXiv preprint arXiv:1912.12405*, 2019.

[48] A. Dahou, M. A. Elaziz, J. Zhou, and S. Xiong, “Arabic sentiment classification using convolutional neural network and differential evolution algorithm,” *Computational intelligence and neuroscience*, vol. 2019, 2019.

[49] H. Shu and Y. Wang, “Automatically searching for u-net image translator architecture,” *arXiv preprint arXiv:2002.11581*, 2020.

[50] L. Li, L. Qin, X. Qu, J. Zhang, Y. Wang, and B. Ran, “Day-ahead traffic flow forecasting based on a deep belief network optimized by the multi-objective particle swarm algorithm,” *Knowledge-Based Systems*, vol. 172, pp. 1–14, 2019.

[51] S. R. Saufi, Z. A. bin Ahmad, M. S. Leong, and M. H. Lim, “Differential evolution optimization for resilient stacked sparse autoencoder and its applications on bearing fault diagnosis,” *Measurement Science and Technology*, vol. 29, no. 12, p. 125002, 2018.

[52] D. Kang and C. W. Ahn, “Efficient neural network space with genetic search,” in *International Conference on Bio-Inspired Computing: Theories and Applications*. Springer, 2019, pp. 638–646.

[53] B. Cheung and C. Sable, “Hybrid evolution of convolutional networks,” in *2011 10th International Conference on Machine Learning and Applications and Workshops*, vol. 1. IEEE, 2011, pp. 293–297.

[54] C. Zhang, P. Lim, A. K. Qin, and K. C. Tan, “Multiobjective deep belief networks ensemble for remaining useful life estimation in prognostics,” *IEEE transactions on neural networks and learning systems*, vol. 28, no. 10, pp. 2306–2318, 2016.

[55] F. Ye, “Particle swarm optimization-based automatic parameter selection for deep neural networks and its applications in large-scale and high-dimensional data,” *PloS one*, vol. 12, no. 12, 2017.

[56] S. Fujino, N. Mori, and K. Matsumoto, “Recognizing the order of four-scene comics by evolutionary deep learning,” in *International Symposium on Distributed Computing and Artificial Intelligence*. Springer, 2018, pp. 136–144.

[57] T. Tanaka, T. Moriya, T. Shinozaki, S. Watanabe, T. Hori, and K. Duh, “Automated structure discovery and parameter tuning of neural network language model based on evolution strategy,” in *2016 IEEE Spoken Language Technology Workshop (SLT)*. IEEE, 2016, pp. 665–671.

[58] A. Kwasigroch, M. Grochowski, and M. Mikolajczyk, “Deep neural network architecture search using network morphism,” in *2019 24th International Conference on Methods and Models in Automation and Robotics (MMAR)*. IEEE, 2019, pp. 30–35.

[59] H. Zhu, Z. An, C. Yang, K. Xu, E. Zhao, and Y. Xu, “Eena: efficient evolution of neural architecture,” in *Proceedings of the IEEE International Conference on Computer Vision Workshops*, 2019, pp. 0–0.

[60] F. M. Johner and J. Wassner, “Efficient evolutionary architecture search for cnn optimization on gtsrb,” in *2019 18th IEEE International Conference On Machine Learning And Applications (ICMLA)*. IEEE, 2019, pp. 56–61.

[61] M. Shen, K. Han, C. Xu, and Y. Wang, “Searching for accurate binary neural architectures,” in *Proceedings of the IEEE International Conference on Computer Vision Workshops*, 2019, pp. 0–0.

[62] Y. Bi, B. Xue, and M. Zhang, “An evolutionary deep learning approach using genetic programming with convolution operators for image

classification," in *2019 IEEE Congress on Evolutionary Computation (CEC)*. IEEE, 2019, pp. 3197–3204.

[63] S. Gibb, H. M. La, and S. Louis, "A genetic algorithm for convolutional network structure optimization for concrete crack detection," in *2018 IEEE Congress on Evolutionary Computation (CEC)*. IEEE, 2018, pp. 1–8.

[64] L. M. Zhang, "A new compensatory genetic algorithm-based method for effective compressed multi-function convolutional neural network model selection with multi-objective optimization," *arXiv preprint arXiv:1906.11912*, 2019.

[65] A. A. Ahmed, S. M. S. Darwish, and M. M. El-Sherbiny, "A novel automatic cnn architecture design approach based on genetic algorithm," in *International Conference on Advanced Intelligent Systems and Informatics*. Springer, 2019, pp. 473–482.

[66] H. Tian, S.-C. Chen, M.-L. Shyu, and S. Rubin, "Automated neural network construction with similarity sensitive evolutionary algorithms," in *2019 IEEE 20th International Conference on Information Reuse and Integration for Data Science (IRI)*. IEEE, 2019, pp. 283–290.

[67] S. Wei, S. Zou, F. Liao, W. Lang, and W. Wu, "Automatic modulation recognition using neural architecture search," in *2019 International Conference on High Performance Big Data and Intelligent Systems (HPBD&IS)*. IEEE, 2019, pp. 151–156.

[68] P. Ortego, A. Diez-Olivan, J. Del Ser, F. Veiga, M. Penalva, and B. Sierra, "Evolutionary lstm-fcn networks for pattern classification in industrial processes," *Swarm and Evolutionary Computation*, vol. 54, p. 100650, 2020.

[69] A. Piergiovanni, A. Angelova, A. Toshev, and M. S. Ryoo, "Evolving space-time neural architectures for videos," in *Proceedings of the IEEE International Conference on Computer Vision*, 2019, pp. 1793–1802.

[70] J. Prellberg and O. Kramer, "Lamarckian evolution of convolutional neural networks," in *International Conference on Parallel Problem Solving from Nature*. Springer, 2018, pp. 424–435.

[71] S. R. Young, D. C. Rose, T. P. Karnowski, S.-H. Lim, and R. M. Patton, "Optimizing deep learning hyper-parameters through an evolutionary algorithm," in *Proceedings of the Workshop on Machine Learning in High-Performance Computing Environments*, 2015, pp. 1–5.

[72] M. G. B. Calisto and S. K. Lai-Yuen, "Self-adaptive 2d-3d ensemble of fully convolutional networks for medical image segmentation," in *Medical Imaging 2020: Image Processing*, vol. 11313. International Society for Optics and Photonics, 2020, p. 113131W.

[73] F. E. F. Junior and G. G. Yen, "Particle swarm optimization of deep neural networks architectures for image classification," *Swarm and Evolutionary Computation*, vol. 49, pp. 62–74, 2019.

[74] B. Wang, Y. Sun, B. Xue, and M. Zhang, "A hybrid differential evolution approach to designing deep convolutional neural networks for image classification," in *Australasian Joint Conference on Artificial Intelligence*. Springer, 2018, pp. 237–250.

[75] L. Frachon, W. Pang, and G. M. Coghill, "Immunecs: Neural committee search by an artificial immune system," *arXiv preprint arXiv:1911.07729*, 2019.

[76] A. I. Sharaf and E.-S. F. Radwan, "An automated approach for developing a convolutional neural network using a modified firefly algorithm for image classification," in *Applications of Firefly Algorithm and its Variants*. Springer, 2020, pp. 99–118.

[77] N. R. Sabar, A. Turky, A. Song, and A. Sattar, "An evolutionary hyper-heuristic to optimise deep belief networks for image reconstruction," *Applied Soft Computing*, p. 105510, 2019.

[78] D. Sapra and A. D. Pimentel, "An evolutionary optimization algorithm for gradually saturating objective functions," in *Proceedings of the Genetic and Evolutionary Computation Conference*, 2020.

[79] A. Kwasigroch, M. Grochowski, and A. Mikołajczyk, "Neural architecture search for skin lesion classification," *IEEE Access*, vol. 8, pp. 9061–9071, 2020.

[80] C. Schorn, T. Elsken, S. Vogel, A. Runge, A. Guntoro, and G. Ascheid, "Automated design of error-resilient and hardware-efficient deep neural networks," *arXiv preprint arXiv:1909.13844*, 2019.

[81] F. Assunção, J. Correia, R. Conceição, M. J. M. Pimenta, B. Tomé, N. Lourenço, and P. Machado, "Automatic design of artificial neural networks for gamma-ray detection," *IEEE Access*, vol. 7, pp. 110 531–110 540, 2019.

[82] D. Laredo, Y. Qin, O. Schütze, and J.-Q. Sun, "Automatic model selection for neural networks," *arXiv preprint arXiv:1905.06010*, 2019.

[83] M. U. Anwar and M. L. Ali, "Boosting neuro evolutionary techniques for speech recognition," in *2019 International Conference on Electrical, Computer and Communication Engineering (ECCE)*. IEEE, 2019, pp. 1–5.

[84] S. Y. Teng, A. C. M. Loy, W. D. Leong, B. S. How, B. L. F. Chin, and V. Máša, "Catalytic thermal degradation of chlorella vulgaris: Evolving deep neural networks for optimization," *Bioresource technology*, vol. 292, p. 121971, 2019.

[85] S. Litzinger, A. Klos, and W. Schiffmann, "Compute-efficient neural network architecture optimization by a genetic algorithm," in *International Conference on Artificial Neural Networks*. Springer, 2019, pp. 387–392.

[86] D. Sapra and A. D. Pimentel, "Constrained evolutionary piecemeal training to design convolutional neural networks," in *International Conference on Industrial, Engineering and Other Applications of Applied Intelligent Systems. Springer*, 2020.

[87] F. Martínez-Álvarez, G. Asencio-Cortés, J. Torres, D. Gutiérrez-Avilés, L. Melgar-García, R. Pérez-Chacón, C. Rubio-Escudero, J. Riquelme, and A. Troncoso, "Coronavirus optimization algorithm: A bioinspired metaheuristic based on the covid-19 propagation model," *arXiv preprint arXiv:2003.13633*, 2020.

[88] E. Rapaport, O. Shriki, and R. Puzis, "Eegnas: Neural architecture search for electroencephalography data analysis and decoding," in *International Workshop on Human Brain and Artificial Intelligence*. Springer, 2019, pp. 3–20.

[89] L. Peng, S. Liu, R. Liu, and L. Wang, "Effective long short-term memory with differential evolution algorithm for electricity price prediction," *Energy*, vol. 162, pp. 1301–1314, 2018.

[90] B. Dahal and J. Zhan, "Effective mutation and recombination for evolving convolutional networks," in *Proceedings of the 3rd International Conference on Applications of Intelligent Systems*, 2020, pp. 1–6.

[91] J. Jiang, F. Han, Q. Ling, J. Wang, T. Li, and H. Han, "Efficient network architecture search via multiobjective particle swarm optimization based on decomposition," *Neural Networks*, vol. 123, pp. 305–316, 2020.

[92] J. Ren, Z. Li, J. Yang, N. Xu, T. Yang, and D. J. Foran, "Eigen: Ecologically-inspired genetic approach for neural network structure searching from scratch," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 9059–9068.

[93] C.-C. Chung, W.-T. Lin, R. Zhang, K.-W. Liang, and P.-C. Chang, "Emotion estimation by joint facial expression and speech tonality using evolutionary deep learning structures," in *2019 IEEE 8th Global Conference on Consumer Electronics (GCCE)*. IEEE, 2019, pp. 221–224.

[94] A. Almalaq and J. J. Zhang, "Evolutionary deep learning-based energy consumption prediction for buildings," *IEEE Access*, vol. 7, pp. 1520–1531, 2018.

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

[96] B. Wang, Y. Sun, B. Xue, and M. Zhang, "Evolving deep convolutional neural networks by variable-length particle swarm optimization for image classification," in *2018 IEEE Congress on Evolutionary Computation (CEC)*. IEEE, 2018, pp. 1–8.

[97] M. P. Wang, "Evolving knowledge and structure through evolution-based neural architecture search," Master's thesis, NTNU, 2019.

[98] F. Assunção, N. Lourenço, P. Machado, and B. Ribeiro, "Fast denser: Efficient deep neuroevolution," in *European Conference on Genetic Programming*. Springer, 2019, pp. 197–212.

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

[100] Z. Lu, I. Whalen, Y. Dhebar, K. Deb, E. Goodman, W. Banzhaf, and V. N. Boddeti, "Multi-criterion evolutionary design of deep convolutional neural networks," *arXiv preprint arXiv:1912.01369*, 2019.

[101] P. R. Lorenzo, J. Nalepa, M. Kawulok, L. S. Ramos, and J. R. Pastor, "Particle swarm optimization for hyper-parameter selection in deep neural networks," in *Proceedings of the genetic and evolutionary computation conference*, 2017, pp. 481–488.

[102] V. Passricha and R. K. Aggarwal, "Pso-based optimized cnn for hindi asr," *International Journal of Speech Technology*, vol. 22, no. 4, pp. 1123–1133, 2019.

[103] H. Zhang, Y. Jin, R. Cheng, and K. Hao, "Sampled training and node inheritance for fast evolutionary neural architecture search," *arXiv preprint arXiv:2003.11613*, 2020.

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

[105] E. Byla and W. Pang, "Deepswarm: Optimising convolutional neural networks using swarm intelligence," in *UK Workshop on Computational Intelligence*. Springer, 2019, pp. 119–130.

[106] M. Baldeon-Calisto and S. K. Lai-Yuen, "Adaresu-net: Multiobjective adaptive convolutional neural network for medical image segmentation," *Neurocomputing*, 2019.

[107] Z. Chen, Y. Zhou, and Z. Huang, "Auto-creation of effective neural network architecture by evolutionary algorithm and resnet for image classification," in *2019 IEEE International Conference on Systems, Man and Cybernetics (SMC)*. IEEE, 2019, pp. 3895–3900.

[108] B. Wang, Y. Sun, B. Xue, and M. Zhang, "Evolving deep neural networks by multi-objective particle swarm optimization for image classification," in *Proceedings of the Genetic and Evolutionary Computation Conference*, 2019, pp. 490–498.

[109] B. Fielding and L. Zhang, "Evolving image classification architectures with enhanced particle swarm optimisation," *IEEE Access*, vol. 6, pp. 68 560–68 575, 2018.

[110] T. Cetto, J. Byrne, X. Xu, and D. Moloney, "Size/accuracy trade-off in convolutional neural networks: An evolutionary approach," in *INNS Big Data and Deep Learning conference*. Springer, 2019, pp. 17–26.

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

[112] M. Loni, A. Majd, A. Loni, M. Daneshtalab, M. Sjödin, and E. Troubitsyna, "Designing compact convolutional neural network for embedded stereo vision systems," in *2018 IEEE 12th International Symposium on Embedded Multicore/Many-core Systems-on-Chip (MCSoC)*. IEEE, 2018, pp. 244–251.

[113] D. Song, C. Xu, X. Jia, Y. Chen, C. Xu, and Y. Wang, "Efficient residual dense block search for image super-resolution," *arXiv preprint arXiv:1909.11409*, 2019.

[114] M. Loni, S. Sinaei, A. Zoljodi, M. Daneshtalab, and M. Sjödin, "Deepmaker: A multi-objective optimization framework for deep neural networks in embedded systems," *Microprocessors and Microsystems*, p. 102989, 2020.

[115] M. Suganuma, M. Kobayashi, S. Shirakawa, and T. Nagao, "Evolution of deep convolutional neural networks using cartesian genetic programming," *Evolutionary Computation*, vol. 28, no. 1, pp. 141–163, 2020.

[116] R. Miikkulainen, J. Liang, E. Meyerson, A. Rawal, D. Fink, O. Francon, B. Raju, H. Shahrzad, A. Navruzyan, N. Duffy *et al.*, "Evolving deep neural networks," in *Artificial Intelligence in the Age of Neural Networks and Brain Computing*. Elsevier, 2019, pp. 293–312.

[117] T. Hassanzadeh, D. Essam, and R. Sarker, "Evou-net: an evolutionary deep fully convolutional neural network for medical image segmentation," in *Proceedings of the 35th Annual ACM Symposium on Applied Computing*, 2020, pp. 181–189.

[118] E. Miahi, S. A. Mirroshandel, and A. Nasr, "Genetic neural architecture search for automatic assessment of human sperm images," *arXiv preprint arXiv:1909.09432*, 2019.

[119] N. Mitschke, M. Heizmann, K.-H. Noffz, and R. Wittmann, "Gradient based evolution to optimize the structure of convolutional neural networks," in *2018 25th IEEE International Conference on Image Processing (ICIP)*. IEEE, 2018, pp. 3438–3442.

[120] X. Chu, B. Zhang, R. Xu, and H. Ma, "Multi-objective reinforced evolution in mobile neural architecture search," *arXiv preprint arXiv:1901.01074*, 2019.

[121] Y. Chen, G. Meng, Q. Zhang, S. Xiang, C. Huang, L. Mu, and X. Wang, "Reinforced evolutionary neural architecture search," *arXiv preprint arXiv:1808.00193*, 2018.

[122] Y. Chen, T. Pan, C. He, and R. Cheng, "Efficient evolutionary deep neural architecture search (nas) by noisy network morphism mutation," in *International Conference on Bio-Inspired Computing: Theories and Applications*. Springer, 2019, pp. 497–508.

[123] Z. Fan, J. Wei, G. Zhu, J. Mo, and W. Li, "Evolutionary neural architecture search for retinal vessel segmentation," *arXiv*, pp. arXiv–2001, 2020.

[124] K. Chen and W. Pang, "Immunetnas: An immune-network approach for searching convolutional neural network architectures," *arXiv preprint arXiv:2002.12704*, 2020.

[125] P. Liu, M. D. El Basha, Y. Li, Y. Xiao, P. C. Sanelli, and R. Fang, "Deep evolutionary networks with expedited genetic algorithms for medical image denoising," *Medical image analysis*, vol. 54, pp. 306–315, 2019.

[126] M. Wistuba, "Deep learning architecture search by neuro-cell-based evolution with function-preserving mutations," in *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer, 2018, pp. 243–258.

[127] O. Kramer, "Evolution of convolutional highway networks," in *International Conference on the Applications of Evolutionary Computation*. Springer, 2018, pp. 395–404.

[128] X. Chu, B. Zhang, H. Ma, R. Xu, J. Li, and Q. Li, "Fast, accurate and lightweight super-resolution with neural architecture search," *arXiv preprint arXiv:1901.07261*, 2019.

[129] Y. Chen, G. Meng, Q. Zhang, X. Zhang, L. Song, S. Xiang, and C. Pan, "Joint neural architecture search and quantization," *arXiv preprint arXiv:1811.09426*, 2018.

[130] B. Wang, B. Xue, and M. Zhang, "Particle swarm optimisation for evolving deep neural networks for image classification by evolving and stacking transferable blocks," *arXiv preprint arXiv:1907.12659*, 2019.

[131] C. Saltori, S. Roy, N. Sebe, and G. Iacca, "Regularized evolutionary algorithm for dynamic neural topology search," in *International Conference on Image Analysis and Processing*. Springer, 2019, pp. 219–230.

[132] T. Wu, J. Shi, D. Zhou, Y. Lei, and M. Gong, "A multi-objective particle swarm optimization for neural networks pruning," in *2019 IEEE Congress on Evolutionary Computation (CEC)*. IEEE, 2019, pp. 570–577.

[133] Y. Hu, S. Sun, J. Li, X. Wang, and Q. Gu, "A novel channel pruning method for deep neural network compression," *arXiv preprint arXiv:1805.11394*, 2018.

[134] P. R. Lorenzo and J. Nalepa, "Memetic evolution of deep neural networks," in *Proceedings of the Genetic and Evolutionary Computation Conference*, 2018, pp. 505–512.

[135] A. ElSaid, F. El Jamiy, J. Higgins, B. Wild, and T. Desell, "Optimizing long short-term memory recurrent neural networks using ant colony optimization to predict turbine engine vibration," *Applied Soft Computing*, vol. 73, pp. 969–991, 2018.

[136] Z. Yang, Y. Wang, X. Chen, B. Shi, C. Xu, C. Xu, Q. Tian, and C. Xu, "Cars: Continuous evolution for efficient neural architecture search," *arXiv preprint arXiv:1909.04977*, 2019.

[137] T. Desell, S. Clachar, J. Higgins, and B. Wild, "Evolving deep recurrent neural networks using ant colony optimization," in *European Conference on Evolutionary Computation in Combinatorial Optimization*. Springer, 2015, pp. 86–98.

[138] Z. Guo, X. Zhang, H. Mu, W. Heng, Z. Liu, Y. Wei, and J. Sun, "Single path one-shot neural architecture search with uniform sampling," *arXiv preprint arXiv:1904.00420*, 2019.

[139] T. Shinozaki and S. Watanabe, "Structure discovery of deep neural network based on evolutionary algorithms," in *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2015, pp. 4979–4983.

[140] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 1–9.

[141] W. Irwin-Harris, Y. Sun, B. Xue, and M. Zhang, "A graph-based encoding for evolutionary convolutional neural network architecture design," in *2019 IEEE Congress on Evolutionary Computation (CEC)*. IEEE, 2019, pp. 546–553.

[142] H. Kitano, "Designing neural networks using genetic algorithms with graph generation system," *Complex systems*, vol. 4, no. 4, pp. 461–476, 1990.

[143] D. Jones, A. Schroeder, and G. Nitschke, "Evolutionary deep learning to identify galaxies in the zone of avoidance," *arXiv preprint arXiv:1903.07461*, 2019.

[144] K. Ho, A. Gilbert, H. Jin, and J. Collomosse, "Neural architecture search for deep image prior," *arXiv preprint arXiv:2001.04776*, 2020.

[145] J. D. Lamos-Sweeney, "Deep learning using genetic algorithms," Ph.D. dissertation, Rochester Institute of Technology, 2012.

[146] A. ElSaid, S. Benson, S. Patwardhan, D. Stadem, and T. Desell, "Evolving recurrent neural networks for time series data prediction of coal plant parameters," in *International Conference on the Applications of Evolutionary Computation (Part of EvoStar)*. Springer, 2019, pp. 488–503.

[147] Y. Sun, G. G. Yen, and Z. Yi, "Evolving unsupervised deep neural networks for learning meaningful representations," *IEEE Transactions on Evolutionary Computation*, vol. 23, no. 1, pp. 89–103, 2018.

[148] A. Behjat, S. Chidambaran, and S. Chowdhury, "Adaptive genomic evolution of neural network topologies (agent) for state-to-action mapping in autonomous agents," in *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, 2019, pp. 9638–9644.

[149] B. Evans, H. Al-Sahaf, B. Xue, and M. Zhang, "Evolutionary deep learning: A genetic programming approach to image classification," in

2018 IEEE Congress on Evolutionary Computation (CEC). IEEE, 2018, pp. 1–6.

[150] S. Bianco, M. Buzzelli, G. Ciocca, and R. Schettini, "Neural architecture search for image saliency fusion," *Information Fusion*, vol. 57, pp. 89–101, 2020.

[151] P. J. Angeline, G. M. Saunders, and J. B. Pollack, "An evolutionary algorithm that constructs recurrent neural networks," *IEEE transactions on Neural Networks*, vol. 5, no. 1, pp. 54–65, 1994.

[152] A. Rawal and R. Miikkulainen, "From nodes to networks: Evolving recurrent neural networks," *arXiv preprint arXiv:1803.04439*, 2018.

[153] M. Märtens and D. Izzo, "Neural network architecture search with differentiable cartesian genetic programming for regression," in *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, 2019, pp. 181–182.

[154] M. Neshat, M. M. Nezhad, E. Abbasnejad, L. B. Tjernberg, D. A. Garcia, B. Alexander, and M. Wagner, "An evolutionary deep learning method for short-term wind speed prediction: A case study of the lillgrund offshore wind farm," *arXiv preprint arXiv:2002.09106*, 2020.

[155] T. Tanaka, T. Shinozaki, S. Watanabe, and T. Hori, "Evolution strategy based neural network optimization and lstm language model for robust speech recognition," *Cit. on*, vol. 130, 2016.

[156] A. Camero, J. Toutouh, J. Ferrer, and E. Alba, "Waste generation prediction under uncertainty in smart cities through deep neuroevolution," *Revista Facultad de Ingeniería Universidad de Antioquia*, no. 93, pp. 128–138, 2019.

[157] I. Loshchilov and F. Hutter, "Cma-es for hyperparameter optimization of deep neural networks," *arXiv preprint arXiv:1604.07269*, 2016.

[158] A. A. ElSaid, A. G. Ororbia, and T. J. Desell, "The ant swarm neuro-evolution procedure for optimizing recurrent networks," *arXiv preprint arXiv:1909.11849*, 2019.

[159] A. ElSaid, F. E. Jamiy, J. Higgins, B. Wild, and T. Desell, "Using ant colony optimization to optimize long short-term memory recurrent neural networks," in *Proceedings of the Genetic and Evolutionary Computation Conference*, 2018, pp. 13–20.

[160] Z. Gao, Y. Li, Y. Yang, X. Wang, N. Dong, and H.-D. Chiang, "A gpso-optimized convolutional neural networks for eeg-based emotion recognition," *Neurocomputing*, vol. 380, pp. 225–235, 2020.

[161] C. Zhang, H. Shao, and Y. Li, "Particle swarm optimisation for evolving artificial neural network," in *Smc 2000 conference proceedings. 2000 ieee international conference on systems, man and cybernetics.'cybernetics evolving to systems, humans, organizations, and their complex interactions'(cat. no. 0*, vol. 4. IEEE, 2000, pp. 2487–2490.

[162] B. P. Evans, H. Al-Sahaf, B. Xue, and M. Zhang, "Genetic programming and gradient descent: A memetic approach to binary image classification," *arXiv preprint arXiv:1909.13030*, 2019.

[163] I. De Falco, G. De Pietro, A. Della Cioppa, G. Sannino, U. Scafuri, and E. Tarantino, "Evolution-based configuration optimization of a deep neural network for the classification of obstructive sleep apnea episodes," *Future Generation Computer Systems*, vol. 98, pp. 377–391, 2019.

[164] J. Huang, W. Sun, and L. Huang, "Deep neural networks compression learning based on multiobjective evolutionary algorithms," *Neurocomputing*, vol. 378, pp. 260–269, 2020.

[165] Z. Lu, I. Whalen, V. Boddeti, Y. Dhebar, K. Deb, E. Goodman, and W. Banzhaf, "Nsga-net: neural architecture search using multi-objective genetic algorithm," in *Proceedings of the Genetic and Evolutionary Computation Conference*, 2019, pp. 419–427.

[166] H. Zhu and Y. Jin, "Real-time federated evolutionary neural architecture search," *arXiv preprint arXiv:2003.02793*, 2020.

[167] J. Dong, A. Cheng, D. Juan, W. Wei, and M. Sun, "Ppp-net: Platform-aware progressive search for pareto-optimal neural architectures," in *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Workshop Track Proceedings*. OpenReview.net, 2018. [Online]. Available: https://openreview.net/forum?id=B1NT3TAIM

[168] D. Hossain and G. Capi, "Multiobjective evolution of deep learning parameters for robot manipulator object recognition and grasping," *Advanced Robotics*, vol. 32, no. 20, pp. 1090–1101, 2018.

[169] J. Bayer, D. Wierstra, J. Togelius, and J. Schmidhuber, "Evolving memory cell structures for sequence learning," in *International Conference on Artificial Neural Networks*. Springer, 2009, pp. 755–764.

[170] S. Roy and N. Chakraborti, "Development of an evolutionary deep neural net for materials research," in *TMS 2020 149th Annual Meeting & Exhibition Supplemental Proceedings*. Springer, 2020, pp. 817–828.

[171] J. Liang, E. Meyerson, B. Hodjat, D. Fink, K. Mutch, and R. Miikkulainen, "Evolutionary neural automl for deep learning," in *Proceedings of the Genetic and Evolutionary Computation Conference*, 2019, pp. 401–409.

[172] J. An, H. Xiong, J. Ma, J. Luo, and J. Huan, "Stylenas: An empirical study of neural architecture search to uncover surprisingly fast end-to-end universal style transfer networks," *arXiv preprint arXiv:1906.02470*, 2019.

[173] M. Javaheripi, M. Samragh, T. Javidi, and F. Koushanfar, "Genecai: Genetic evolution for acquiring compact ai," *arXiv preprint arXiv:2004.04249*, 2020.

[174] K. Deb, *Multi-objective optimization using evolutionary algorithms*. John Wiley & Sons, 2001, vol. 16.

[175] K. Maziarz, A. Khorlin, Q. de Laroussilhe, and A. Gesmundo, "Evolutionary-neural hybrid agents for architecture search," *CoRR*, vol. abs/1811.09828, 2018. [Online]. Available: http://arxiv.org/abs/1811.09828

[176] T. Chen, I. Goodfellow, and J. Shlens, "Net2net: Accelerating learning via knowledge transfer," *arXiv preprint arXiv:1511.05641*, 2015.

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

[178] M. Mitchell, *An introduction to genetic algorithms*. MIT press, 1998.

[179] K. Deb, R. B. Agrawal *et al.*, "Simulated binary crossover for continuous search space," *Complex systems*, vol. 9, no. 2, pp. 115–148, 1995.

[180] D. V. Vargas and S. Kotyan, "Evolving robust neural architectures to defend from adversarial attacks," *arXiv preprint arXiv:1906.11667*, 2019.

[181] K. Deb, "Multi-objective optimization," in *Search methodologies*. Springer, 2014, pp. 403–449.

[182] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: Nsga-ii," *IEEE transactions on evolutionary computation*, vol. 6, no. 2, pp. 182–197, 2002.

[183] Q. Zhang and H. Li, "Moea/d: A multiobjective evolutionary algorithm based on decomposition," *IEEE Transactions on evolutionary computation*, vol. 11, no. 6, pp. 712–731, 2007.

[184] K. Deb and H. Jain, "An evolutionary many-objective optimization algorithm using reference-point-based nondominated sorting approach, part i: solving problems with box constraints," *IEEE transactions on evolutionary computation*, vol. 18, no. 4, pp. 577–601, 2013.

[185] C. A. C. Coello, G. T. Pulido, and M. S. Lechuga, "Handling multiple objectives with particle swarm optimization," *IEEE Transactions on evolutionary computation*, vol. 8, no. 3, pp. 256–279, 2004.

[186] M. R. Sierra and C. A. C. Coello, "Improving pso-based multi-objective optimization using crowding, mutation and $\in$-dominance," in *International conference on evolutionary multi-criterion optimization*. Springer, 2005, pp. 505–519.

[187] B. Wang, Y. Sun, B. Xue, and M. Zhang, "A hybrid ga-pso method for evolving architecture and short connections of deep convolutional neural networks," in *Pacific Rim International Conference on Artificial Intelligence*. Springer, 2019, pp. 650–663.

[188] D. R. So, C. Liang, and Q. V. Le, "The evolved transformer," *arXiv preprint arXiv:1901.11117*, 2019.

[189] F. Assunção, N. Lourenço, P. Machado, and B. Ribeiro, "Evolving the topology of large scale deep neural networks," in *European Conference on Genetic Programming*. Springer, 2018, pp. 19–34.

[190] F. Assunçao, N. Lourenço, P. Machado, and B. Ribeiro, "Denser: deep evolutionary network structured representation," *Genetic Programming and Evolvable Machines*, vol. 20, no. 1, pp. 5–35, 2019.

[191] J. Liu, M. Gong, Q. Miao, X. Wang, and H. Li, "Structure learning for deep neural networks based on multiobjective optimization," *IEEE transactions on neural networks and learning systems*, vol. 29, no. 6, pp. 2450–2463, 2017.

[192] G. Bender, P. Kindermans, B. Zoph, V. Vasudevan, and Q. V. Le, "Understanding and simplifying one-shot architecture search," in *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, ser. Proceedings of Machine Learning Research, J. G. Dy and A. Krause, Eds., vol. 80. PMLR, 2018, pp. 549–558. [Online]. Available: http://proceedings.mlr.press/v80/bender18a.html

[193] X. Chu, B. Zhang, R. Xu, and J. Li, "Fairnas: Rethinking evaluation fairness of weight sharing neural architecture search," *arXiv preprint arXiv:1907.01845*, 2019.

[194] P. Colangelo, O. Segal, A. Speicher, and M. Margala, "Artificial neural network and accelerator co-design using evolutionary algorithms," in *2019 IEEE High Performance Extreme Computing Conference (HPEC)*. IEEE, 2019, pp. 1–8.

[195] ——, "Evolutionary cell aided design for neural network architectures," *arXiv preprint arXiv:1903.02130*, 2019.

[196] Y. Sun, H. Wang, B. Xue, Y. Jin, G. G. Yen, and M. Zhang, "Surrogate-assisted evolutionary deep learning using an end-to-end random forest-based performance predictor," *IEEE Transactions on Evolutionary Computation*, 2019.

[197] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[198] S. S. Tirumala, "Evolving deep neural networks using coevolutionary algorithms with multi-population strategy," *Neural Computing and Applications*, pp. 1–14, 2020.

[199] F. Assunção, N. Lourenço, P. Machado, and B. Ribeiro, "Fast-denser++: Evolving fully-trained deep artificial neural networks," *arXiv preprint arXiv:1905.02969*, 2019.

[200] F. Assunçao, N. Lourenço, B. Ribeiro, and P. Machado, "Incremental evolution and development of deep artificial neural networks," in *Genetic Programming: 23rd European Conference, EuroGP 2020, Held as Part of EvoStar 2020, Seville, Spain, April 15–17, 2020, Proceedings 23*. Springer, 2020, pp. 35–51.

[201] C. Wei, C. Niu, Y. Tang, and J. Liang, "Npenas: Neural predictor guided evolution for neural architecture search," *arXiv preprint arXiv:2003.12857*, 2020.

[202] J. Song, Y. Jin, Y. Li, and C. Lang, "Learning structural similarity with evolutionary-gan: A new face de-identification method," in *2019 6th International Conference on Behavioral, Economic and Socio-Cultural Computing (BESC)*. IEEE, 2019, pp. 1–6.

[203] L. M. Zhang, "Genetic deep neural networks using different activation functions for financial data mining," in *2015 IEEE International Conference on Big Data (Big Data)*. IEEE, 2015, pp. 2849–2851.

[204] J. Liang, E. Meyerson, and R. Miikkulainen, "Evolutionary architecture search for deep multitask networks," in *Proceedings of the Genetic and Evolutionary Computation Conference*, 2018, pp. 466–473.

[205] S. Fujino, T. Hatanaka, N. Mori, and K. Matsumoto, "The evolutionary deep learning based on deep convolutional neural network for the anime storyboard recognition," in *International Symposium on Distributed Computing and Artificial Intelligence*. Springer, 2017, pp. 278–285.

[206] J.-D. Dong, A.-C. Cheng, D.-C. Juan, W. Wei, and M. Sun, "Dpp-net: Device-aware progressive search for pareto-optimal neural architectures," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 517–531.

[207] A.-C. Cheng, J.-D. Dong, C.-H. Hsu, S.-H. Chang, M. Sun, S.-C. Chang, J.-Y. Pan, Y.-T. Chen, W. Wei, and D.-C. Juan, "Searching toward pareto-optimal device-aware neural architectures," in *Proceedings of the International Conference on Computer-Aided Design*, 2018, pp. 1–7.

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

[209] K. Yu, C. Sciuto, M. Jaggi, C. Musat, and M. Salzmann, "Evaluating the search phase of neural architecture search," *arXiv preprint arXiv:1902.08142*, 2019.

[210] P. Liashchynskyi and P. Liashchynskyi, "Grid search, random search, genetic algorithm: A big comparison for nas," *arXiv preprint arXiv:1912.06059*, 2019.

[211] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[212] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *2009 IEEE conference on computer vision and pattern recognition*. Ieee, 2009, pp. 248–255.

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

[214] M. D. Zeiler and R. Fergus, "Visualizing and understanding convolutional networks," in *European conference on computer vision*. Springer, 2014, pp. 818–833.

[215] J. R. Koza and J. R. Koza, *Genetic programming: on the programming of computers by means of natural selection*. MIT press, 1992, vol. 1.

[216] F. Cheng, J. Yu, and H. Xiong, "Facial expression recognition in jaffe dataset based on gaussian process classification," *IEEE Transactions on Neural Networks*, vol. 21, no. 10, pp. 1685–1690, 2010.

[217] J. Triesch and C. Von Der Malsburg, "Robust classification of hand postures against complex backgrounds," in *Proceedings of the second international conference on automatic face and gesture recognition*. IEEE, 1996, pp. 170–175.

[218] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling," *arXiv preprint arXiv:1412.3555*, 2014.

[219] G.-B. Zhou, J. Wu, C.-L. Zhang, and Z.-H. Zhou, "Minimal gated unit for recurrent neural networks," *International Journal of Automation and Computing*, vol. 13, no. 3, pp. 226–234, 2016.

[220] J. Collins, J. Sohl-Dickstein, and D. Sussillo, "Capacity and trainability in recurrent neural networks," *arXiv preprint arXiv:1611.09913*, 2016.

[221] B. Zoph, V. Vasudevan, J. Shlens, and Q. V. Le, "Learning transferable architectures for scalable image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 8697–8710.

[222] C. Ying, A. Klein, E. Christiansen, E. Real, K. Murphy, and F. Hutter, "Nas-bench-101: Towards reproducible neural architecture search," in *International Conference on Machine Learning*, 2019, pp. 7105–7114.

[223] X. Dong and Y. Yang, "Nas-bench-201: Extending the scope of reproducible neural architecture search," in *International Conference on Learning Representations (ICLR)*, 2020. [Online]. Available: https://openreview.net/forum?id=HJxyZkBKDr