A Survey of Advances in Evolutionary Neural Architecture Search

1st Xun Zhou
Department of Computer Science
City University of Hong Kong
Kowloon Tong, Hong Kong
xunzhou6-c@my.cityu.edu.hk

2nd A. K. Qin
Department of Computer Science and Software Engineering
Swinburne University of Technology
Hawthorn, Australia
kqin@swin.edu.au

3rd Yanan Sun
College of Computer Science
Sichuan University
Chengdu, China
ysun@scu.edu.cn

4th Kay Chen Tan
Department of Computing
The Hong Kong Polytechnic University
Hung Hom, Hong Kong
kaychen.tan@polyu.edu.hk

2021 IEEE Congress on Evolutionary Computation (CEC) | 978-1-7281-8393-0/21/$31.00 ©2021 IEEE | DOI: 10.1109/CEC45853.2021.9504890

***Abstract*—Deep neural networks (DNNs) have been frequently and widely applied for intelligent systems such as object detection, natural language understanding and speech recognition. Given a specific problem, we always aim to construct the most suitable DNN to solve it, which requires choosing the most appropriate model architecture and seeking the best model parameters values. However, most existing works focus on model parameters learning under the assumption that the model architecture can be manually specified as per prior knowledge and/or trial-and-error experimentation. To overcome this problem, evolutionary algorithms (EAs) have been widely used to design model architectures automatically. Further, EAs have been used for neural network optimization for more than 30 years. Therefore, in this paper, we review the evolutionary neural architecture search (ENAS) from the view of the advanced techniques. We hope this work can provide a comprehensive understanding of EAs' roles for the readers and focus themselves on ENAS.**

***Index Terms*—Evolutionary Algorithms, Neural Architecture Search, Optimization**

I. Introduction

Deep neural networks (DNNs) are representatives methods in machine learning. By expanding to more complex structures of neurons that can extract multi-level features, DNNs are more effective to process regression, classification and reconstruction tasks [1]. Particularly, in recent years, the artificial intelligent systems equipped with DNNs, e.g., machine translation, speech recognition and object detection, have changed the human life thoroughly [2].

The outstanding performance of DNNs is associated with the model architecture design for various data types, including convolutional neural network (CNN) for images, deep belief network (DBN) for data with independent input features and recurrent neural network (RNN) for sequence data [3]. Although the development of DNNs has been continued over decades, most of the state-of-the-art network architectures are still manually designed through hundreds of experts relying on the expensive trial-and-error process [4]. To overcome this problem, neural architecture search (NAS) is proposed for designing the neural networks automatically [5]. NAS can be seen as an optimization problem where finding out the optimal model architecture to achieve the best performance on the specific tasks. Because it is often hard to represent the model architecture by continued parameters and there is no explicit formulation between the model architecture and its performance, this optimization problem is often characterized with non-convex and black-box properties [6], [7].

To solve the optimization problem mentioned above, lots of methods have been proposed during the past years. Specifically, the representatives are reinforcement learning (RL) based methods [8]–[12], gradient-based methods [7], [13], [14] and evolutionary algorithms (EAs). Among these methods, EAs are often utilized in line with the development of neural networks. They played an important role in designing novel ANNs architectures and training them in the 1990s [15]. Although from the 2000s when researchers were gradually paying more attention to other machine learning methods, EAs still made a great contribution to neural networks, and multiple outstanding works had been done such as NeuroEvolution of Augmenting Topologies (NEAT) [16] and HyperNEAT [17]. With the renaissance of deep learning, EAs are applied frequently to optimize the networks with the deep structure. The major reason is that EAs are flexible. There is no constraint for the problems to be solved. It means that EAs are natural for this non-convex and black-box problem [6]. Also, EAs are highly parallelizable, where based on the extensive available computational resources, the optimal architecture can be searched within a short time [18], [19].

This work is partially supported by the National Key Research and Development Project, Ministry of Science and Technology, China (Grant No. 2018AAA0101301), the National Natural Science Foundation of China (NSFC) under grant No. 61876162, the Research Grants Council of the Hong Kong SAR under grant no. PolyU11202418 and PolyU11209219, and the Australian Research Council (ARC) under grant no. LP170100416 and LP180100114.

978-1-7281-8393-0/21/$31.00 ©2021 IEEE

In this paper, from the advances in evolutionary neural architecture search (ENAS), applications of EAs for DNN model architecture optimization are reviewed. Particularly, we focus on the critical techniques of EAs in different periods of time. We hope this work can provide an overview of the ENAS' development for the readers and can be a start point for them to begin the research work in this area.

The remainder of this paper is organized as follows. Section II provides an overview of NAS, including problem statement, existing methods and ENAS. Section III reviews the advanced techniques in ENAS along the timeline. Section IV discusses the performance of related works and potential problems in ENAS. Section V concludes this paper.

II. Neural Architectural Search

As a highly complex optimization problem, NAS has been solved by various kinds of optimization methods in recent years. In this section, by formulating this problem, its properties are summarized at first. Then, the principles of these mainstream methods are shown. Furthermore, the natural advantages of EAs are provided for the readers.

A. Problem statement

Two aspects determine the performance of neural networks, i.e., model architecture M_a and model parameters M_p. The former refers to the basic units in the network, including the unit types (e.g., convolution layer, pooling layer and fully connected layer), number of units and connections among these units, which defines the model's capabilities. The latter is about trainable parameters in these units, i.e., connection weights and biases. Therefore, to design the optimal model for the specific task, NAS requires searching the optimal model architecture associated with the model parameters. It can be formulated as an optimization problem in Equation (1),

$$\begin{cases} \mathscr{P}(M_a, M_p) = \underset{M_a, M_p}{\arg\min}\, O_{D^{val}}(M_a, M_p) \\ s.t.\ \ constraint_i(M_a),\ i = 1, ..., m \\ \qquad M_p = \underset{M'_p}{\arg\min}\, L_{D^{trn}}(M_a, M'_p) \\ \qquad s.t.\ \ constraint_i(M'_p),\ i = 1, ..., k \end{cases} \tag{1}$$

where it is a bi-level optimization problem [7]. The upper-level optimization problem means finding out the optimal model architectures on the Pareto Front[1] to minimize the (maybe) multiple objective functions on the validation dataset D^{val}. The lower-level optimization problem means optimizing the model parameters to minimize the loss function on the training dataset D^{trn}. There are also various constraints:

- One is about limiting the search space for candidate solutions, including predefining the type of possible units [4] and fixing the max number of units in the network [20].
- Another is about preventing the invalid solutions from being found, including the invalid architectures (such as a fully connected layer followed by a convolution layer [21]) and these architectures which don't meet the specific requirements, e.g., model size [22].

From the description above, it can be seen that NAS is a prohibitively challenging optimization problem. Specifically, properties of this problem is summarized as follows:

- a giant search space, where the candidate model architectures are exponentially increasing with the possible connections, number of units and units' types;
- a black-box optimization problem, because the performance of model architecture is unable to be formulated;
- a non-convex and non-differentiable search landscape, because model architecture is hard to be represented by continue parameters, and
- may be a multi-objective optimization problem with various constraints.

B. Existing methods

The optimization problem (1) is often solved iteratively. In this way, an optimizer is designed to sample the candidate model architectures from the search space; then, these candidates are trained on the training dataset by gradient-based methods and evaluated on the validation dataset; Evaluations of each model are as feedback to help the optimizer to reproduce model architectures with the higher performance [23]. There are three mainstream optimization methods applied in the current works, i.e., RL-based methods, EA-based methods, and gradient-based methods.

Reinforcement learning based methods. For the RL-based methods, a policy based on the Markov decision process is design to produce the promising model architectures [8], [9], [23]. Specifically, the policy is applied to construct the DNN gradually, and then the performance of the constructed network is the reward to optimize the policy in turn. The goal of RL-based methods is to optimize a policy to produce DNN with the highest reward. For example, in [23], a RNN is used as the policy. The RNN receives the architecture information of the network and outputs the units or connections to be added in it to generate a larger model step by step. Then, the performance of the produced model is as feedback to optimise the RNN. This procedure is iterative to make the RNN produce the model architecture with the higher accuracy.

Evolutionary algorithms based methods. The evolution and cooperation of biology inspire the design of EAs with the population-based behaviors [24]. Therefore, when EAs are applied for NAS, a population of model architectures is initialized. Then, based on each model's performance, these model architectures are modified by the evolutionary operations, e.g., recombination and mutation, to reproduce the new individuals with the higher performance. In the end, the model with the highest performance is selected as the output. Particularly, in EAs, to design the flexible evolutionary operations, the model architectures are often represented by the special encoding. For

[1]Here NAS is formulated as a multi-objective functions because not only accuracy is considered but also other measurements, e.g., model size and latency, should be considered to construct a model architecture. Also, Pareto optimal set are the solutions for the multi-objective optimization problem.

example, the adjacent binary matrix is often used to represent the connections among units [25].

Gradient-based methods. Considering the search space of model architecture is often discrete and thus if the gradient-based methods are used, this search space should be converted to the continuous one. Then, the gradient information can be calculated to guide the search process. For example, in [7], through relaxing the discrete architecture weights $w_a \in \{0,1\}$ for all the candidate units to the continuous values $w_a \in [0,1]$, gradient information Δw_a can be obtained to update the w_a and after fully training the unit with the highest architecture weight is selected to construct the target network.

C. ENAS

ENAS means applying EAs to search the model architecture automatically. EAs are the population-based methods that are inspired by biological behaviors, including the genetic algorithm (GA), genetic programming (GP), evolution strategy (ES) and differential evolution (DE). All EAs follow the general framework to solve problems, e.g., initialization, evaluation, reproduction and selection. Specifically, a set of model architectures are produced as the initial population. According to the existing works, the usual initial strategy is randomly generating these model architectures. Then, these models are trained and evaluated for the specific task, and these evaluations (e.g., accuracy, model size and latency) are the model's fitness values. Afterward, these evolutionary operations are applied to reproduce the new model architectures based on the population and their fitness values. For example, the recombination operation is used to exchange two model architectures and produces two new models, and the mutation operator is about changing the single model architecture (e.g., adding one skip connection, removing one layer or adjusting one convolution layer's size) to reproduce the new one. Finally, in the selection stage, the elites are selected from the old population and the reproduced model architectures to construct the population for next generation. The procedure mentioned above is iterative unit some termination criteria are met (e.g., reaching the max generation).

EAs have been applied for optimizing DNN for more than 30 years because their direct search procedure makes these algorithms natural for the complex optimization problem. Compared with the RL-based methods, model architectures searched by EAs have the higher accuracy at the initial stage, meaning that EAs are suitable for the computation limitation situation [26]. Also, few meta-parameters of EAs make these algorithms simpler [26]. Compared with the gradient-based methods, although EAs were time-consuming in the early time, with the use of speed-up techniques, such as inheritance [27] and one-shot model [28], the EA-based methods can find out the promising model architecture in a short time. Significantly, the gradient-based methods are with stronger constraints on the search space than the EA-based methods, which means the EA-based methods are more suitable for exploring various kinds of search space, including the infinite one.

Therefore, in this paper, we focus on applications of EAs on NAS, and in the next Section, the typical techniques in different time stages are reviewed.

III. Advances in Evolutionary Architecture Search

A. Shallow networks

In the last century, with the application of gradient-based methods to train neural networks, the outstanding performance of these models was recognized by researchers immediately. However, these gradient-based methods severely suffer from easily trapping into the local minimum due to their local search nature. Considering the global search nature of EAs, these algorithms were used to train neural networks as an alternative method. Besides, compared with gradient-based methods, EAs can be applied to design the model architecture, including the number of neurons, the connections among neurons and neurons' activation functions. Therefore, massive works that optimized model architecture, model parameters and both of them by EAs appeared. Fortunately, Yao [15] had provided a systematic survey for these early works.

Researchers gradually paid more attention to the other machine learning methods such as support vector machine, entering the new century. However, the EA-based methods were still applied to promote the development of neural networks. In 2002, NEAT was proposed by Stanley et al. [16] to design the neural networks more flexibly. Specifically, NEAT followed an incremental paradigm where from a single neuron to a complex network. In this work, to adjust the network architecture flexibly, graph encoding was used. For this encoding, there were two parts, i.e., node genes and connection genes. The former included the information of neurons, including their activation functions. The latter was similar to the adjacent list, which can represent the connections among units with weights. Fig 1 is applied to show the representation of network in NEAT. Based on the graph encoding, the effective recombination can swap architectures between two models and the mutation can be applied to insert or eliminate the neurons or connections in the network flexibly. In 2007, HyperNEAT was proposed [29]. Compared with the original version in [16], HyperNEAT also applied the EAs to design the architecture and optimize weights of neural networks, but it used the indirect encoding to represent the model's weights instead of the real-value encoding, where the model with the larger size parameters can be searched.

Although model architectures can be designed by EAs flexibly, these works mainly focused on the shallow forward neural networks with simple neurons. Limited by the architecture and capabilities of these basic units, these networks are commonly used to deal with simple tasks, such as handwriting recognition.

B. Deep networks

In 2012, AlexNet, one kind of CNN, obtained the highest classification accuracy on the dataset ImageNet, and researchers turned to design the neural networks with the

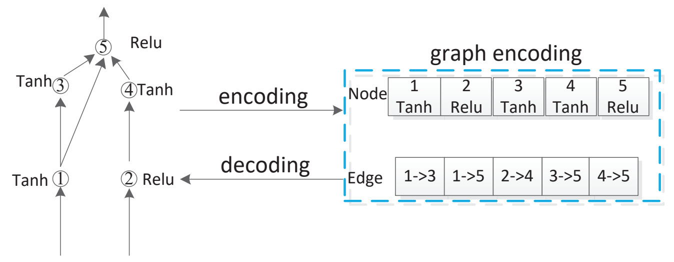

Fig. 1. A network consists of 5 neurons and represented by graph encoding in NEAT.

deeper architecture again [30]. Not surprisingly, EAs played an essential role in DNN architecture optimization. During this time, related work could be divided into two types.

One is optimizing the architecture parameters under the fixed connection pattern, including optimising the number of layers and hyper-parameters of each layer (e.g., the kernel size for a convolution layer and the number of neurons in the fully connected layer). Specific encodings often represented these architecture parameters and EAs were applied to search the optimal code values to make the model achieve the higher performance. A naive representation for the architecture parameters is the array list, where each element corresponds to the layer's hyper-parameters and the length of the list is the number of layers. By adjusting the elements' values and inserting or eliminating elements, DNN with various architectures can be obtained easily. Fig 2 is applied to show a CNN's architecture parameters optimized by EAs. For example, in 2016, Zhang et al. [31] applied the DE to optimize the number of neurons of three hidden layers in a DBN, and Young et al. [32] applied the GA to optimize the kernel size and number of filters of each convolution layer in the CNN. Also, in 2015, Gong et al. [33] through optimizing the bias of each neuron, applied the DE to sparsify the connections between the adjacent fully connected layers and in their follow-up work [34] EA was used to construct the possible connections layer by layer based on the binary representation.

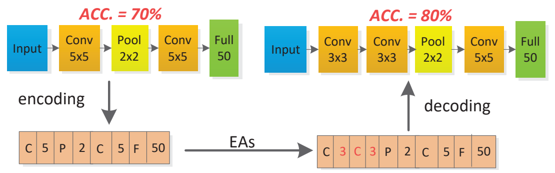

Fig. 2. A simple CNN is optimized by EAs. In the encoding *C*, *P* and *F* denotes the types of the layer as convolution layer, pooling layer and fully connected layer, respectively. Also, ACC. denotes model's accuracy.

The second type was predefining layers and the optimizing connections among them. For example, Shinozaki et al. [25] applied the adjacent matrix to represent the free connections among fully connected layers and GA was used to search the optimal topology of these basic units; Desell et al [35] applied the ant colony to design the connections of neurons in the cell of a deep RNN. A simple example in [25] is shown in Fig. 3 to show the connections optimized by EAs. Considering that most of DNNs are directed acyclic graphes, the adjacent matrix can be compressed as a vector, and Xie et al. [20] applied this compressed vector, which was suitable for the GA, to represent the connections among convolution layers.

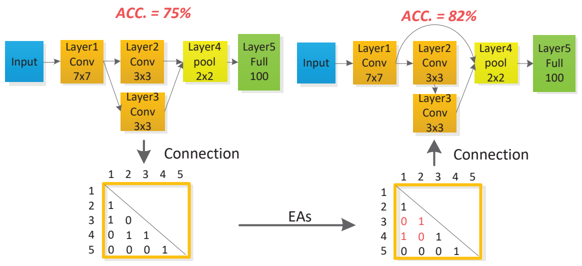

Fig. 3. A CNN's connections is represented by the binary adjacent matrix and optimized by EAs, where all the layers are numbered and 1 in the matrix denotes that a connection exists between these two layers.

Through applying EAs, architecture parameters or the connections which were often adjusted manually can be optimized automatically. As a result, the optimized model architectures had the higher accuracy on the popular benchmark. For example, on ImageNet, compared VGGNet-16 28.5% and AlexNet 42.6% top-1 test error, GeNet [20] optimized by EAs obtain the more accurate result, i.e., 27.87% top-1 test error. Besides, the application of DNNs was promoted further. For example, in [31] by adjusting the neurons number of the traditional DBN, this deep model obtained the highest prediction result on the machine's usable life than lots of predictors, e.g., MLP, SVM and ELM.

However, the application of EAs was still on the traditional way where parts of the networks were predefined manually and the remained parameters were optimized by EAs to make these networks achieve the predictive performance. Therefore, innovation was expected to improve the automatic degree of the model architecture design further.

C. *Innovation in deep networks*

To enhance the degree of automatic search, in 2017, the Google groups applied EAs to search the entire CNN automatically [4]. The method used in this work can be seen as an improved NEAT, where a basic unit replaced each node and only the mutation operation was considered to reproduce new individuals. Researchers only needed to define the possible units in the network and design the mutation operations where a DNN can be obtained fully automatically.

Referring to the outperformance of fully automatically designed DNNs [4], the follow-up works focused on proposing the more effective EAs to search the model architecture with a higher performance. Specifically, the improvement was from two aspects, i.e., model architecture representation and evolutionary operations design. The former makes the search space of model architecture more smooth and can be explored more effectively. For example, Suganuma et al. [36] converted the graph encoding into a list by embedding the adjacent list into the node gene. As a result, the connections and the architecture parameters of units can be searched simultaneously. Liu et al. [37] proposed a hierarchical representation. The network is divided into sub-networks with different levels where the high-level sub-networks consist of the lower ones, and the lowest-level is about a set of basic units, i.e., convolution layer and pooling layer. Mutation can appear in any sub-network to change its architecture. Through this representation, various kinds of CNNs can be searched by the very simple evolutionary operation. Fig. 4 is applied to show the hierarchical representation clearly.

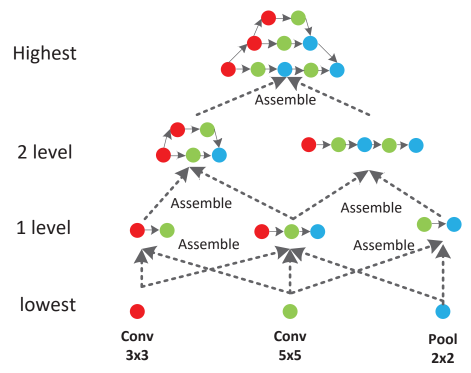

Fig. 4. This network is divided into 4 level, where the lowest is about three basic layer and the higher level subnetwork is constructed by the lower one. The highest level corresponds to the network.

The latter designed the more effective operations to search the optimal model architecture, including the initialization, recombination, mutation and selection. For example, based on professional knowledge, the initial model architectures for EAs were predefined [38], which reduced the search space significantly and can lead the EAs to converge to the optimal model faster. In [39], a specific layer-wise recombination was proposed to swap the architectures between the two models with different depths. For this recombination operation, only the units with the same type and at the same positions can be exchanged. This amended recombination operation was helpful to find the architectures with different depths. The mutation was often applied to explore the new region in search space, but considering that the search space of model architecture was giant, this operation may search in the insignificant regions. Therefore, Chen et al. [40] designed a controller to mutate the architecture to a novel one or to the one with the higher performance, which made a trade-off between exploration and exploitation. Besides, Real et al. [26] also considered the extra attribute *age* in the selection operation, where the older individuals were eliminated from the population in priority. This amended selection pushed the population frequently renewed, which can be seen as a regularization in EAs to prevent these algorithms from trapping into the local optimums.

With the automated model architecture design by EAs proposed and the follow-up improvements, ENAS reached a great achievement. Specifically, as the groundbreaking work [4], the obtained large-scale CNN achieves 4.4% test error on CIFAR-10, which is more accurate than the mainstream hand-crafted models ResNet [41] 6.6% and highway networks [42] 7.7%. For these follow-up works, the accuracy of the automatically designed models was improved further. For example, in [37] the hierarchical representation model reached 3.75% test error on CIFAR-10, RENASNet [40] reached 2.88% test error and AmoebaNet-A [26] reached 3.40% test error.

D. *Booming of deep networks*

With huge efforts, the model architectures designed by EAs achieved state-of-the-art accuracy on many benchmark datasets, including the CIFAR-10, CIFAR-100, and ImageNet. However, the emergency for ENAS is that large computational resources are needed to search an entire network. For example, in [4], around 2700 GPU days were required to search the expected model. Although reducing the search space with constraints, such as predefining the maximum number of layers, possible units and connection pattern, ENAS was still time-consuming. Therefore, researchers were inclined to study the speed-up strategies to search the model architecture much faster with negligible accuracy loss.

The basic search paradigm for ENAS was an iterative procedure, where EAs reproduced new model architectures at first. These models were then trained and evaluated, and further, these evaluations were as feedback to guide the EAs to produce the models with higher performance. During this process, large computational resources were spent on training these models. Therefore, the core step for reducing search cost was how to evaluate the models with the less training.

The simple way is to consider traditional speed-up strategies for the training process, including reducing training data size, reducing training epoch and predicting the learning curves. For NAS, a specific strategy is optimizing a proxy to predict the model's performance. This proxy can be seen as the mapping from the model's architecture to its performance. Through inputting architecture-related data, such as the number of layers, type of these layers and connections, the proxy can output the model's accuracy for the specific task without any training. As a result, the time cost for NAS can be reduced significantly. For example, in [43], the proxy was designed based on random forest, and a set of training data samples consist of the CNN architectures (incl. numbers of DenseNet blocks, ResNet blocks and Pooling blocks) and their accuracies. After the proxy was optimized, the model architecture produced by EAs can be input to this proxy, and then its accuracy was output directly. Compared with training all the candidate architectures during search space, the time cost for training the proxy can be neglected. As a result, the automatically designed CNN with 94.70% accuracy on the CIFAR-10 dataset was competitive with the works without speed up strategies, but the time cost was significant reduced from 2750 GPU days to 8.5 GPU days.

Besides, inheritance is a customized speed-up strategy in EAs, where through inheriting the model parameters from the parents, it is unnecessary to train the generated children from scratch. Specifically, during the reproduction process (e.g., recombination and mutation), the model parameters are kept in the corresponding basic layers as a start point for the children models. Then, with short optimization runs of gradient-based methods, these children can achieve the expected performance and be evaluated on the validation dataset to obtain their fitness values. For example, Elsken et al. [44] designed a simple mutation operation based on the inheritance to search model architecture. In this work, a fully pre-trained simple model was as a start point, and then through adding a block or increasing the channels, the deeper or wider children can be obtained. These children are trained and the one with the higher performance is selected as a parent to reproduce the new individuals. In Fig 5, there is an example about the recombination associated with inheritance to reproduce new architectures. Because there was no need to train the children from scratch, the time cost was reduced to several GPU days (0.5-2 GPU days differs from the model sizes) to obtain the model architectures with about 95% on the CIFAR-10 dataset.

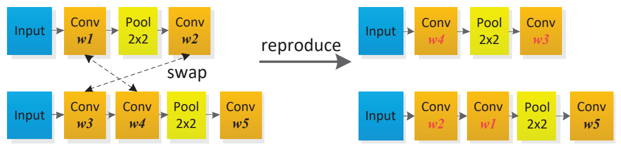

Fig. 5. Two model architectures are swapped with each other and these swapped layers inherit the model parameters w from the parents.

Afterward, the one-shot model came into public attention and was gradually combined with the EA-based methods. The one-shot model is about training a supernet at once and inferring the optimal model architectures from it. Specifically, there are often two stages in this model, i.e., the training stage and deriving stage. A supernet is constructed in the training stage, which contains all the candidate model architectures in the search space and is trained by the gradient optimization methods. Then, in the deriving stage, subnetworks can be clipped from the supernet through inheriting the model parameters, and these networks are evaluated directly. Finally, the model architecture with the highest evaluations is chosen as the optimal solution. In [22], EAs as the search methods were applied in the second stage to clip the optimal model architecture from the supernet. The model architectures found out by EAs can be obtained within 1 GPU day with 74.3% accuracy on the ImageNet dataset from the experiment results.

Almost at the same time, the cell-based architecture was applied to speed up the search process. Instead of searching the entire network, EAs are used to search a cell (also can be seen as a sub-network) and through simply stacking the cell for various times, a completed network can be obtained. Compared with the entire network, the cell's search space is so small that the time cost can be reduced. For example, in [37], a cell was searched by EAs automatically. Through stacking the cell three times, a CNN with the top-1 error of 3.6% on CIFAR-10 can be obtained, and through stacking the same cell seven times, a larger CNN can work on the ImageNet with 20.3% top-1 error. Compared with using 250 GPUs to search a entire network in 11 days, based on the cell-based architecture, the time cost can be reduced to 1.5 days on 200 GPUs. However, compared with other speed-up strategies, the time cost is still high.

With the speed-up strategies mentioned above, the model architectures can be searched by EAs in several days on the single GPU, which promoted the ENAS work by individual researcher. Because of the high degree of automatic search, this technique was attractive for the researchers without enough DNN knowledge to design the DNNs for the specific problems, such as gamma-ray detection [45], crack detection of concrete [46], electricity price prediction [47] and medical images [48].

E. Trends

With the proposed advanced EAs techniques and speed-up strategies, researchers are devoted to searching the model architecture with higher accuracy with few costs [28], [49], [50]. For example, in [28], the one-shot model was combined with the cell-based architecture, where EAs were applied to clip the cell from the supernet. In [49], the inheritance operator and cell-based architecture were used together. In this work, several kinds of cells were searched to construct the model architectures with the higher accuracy and through inheriting the model parameters from the parents models, the time cost was reduced significantly. By merging these advanced techniques elaborately, EAs can currently search the models with very high performance on the benchmark within one day on the single GPU.

On the other hand, researchers are enhancing the degree of automation for NAS further. Although the entire network can be searched quickly, the search space, which denotes what kinds of architectures can be found, is still designed by the experts. For example, in [28] cell consisted of 4 nodes with 8 possible basic operations and in [22] there was a 20 layers network with 4 possible units in each layer. Therefore, EAs were applied to design the search space automatically. In [51], the original search space consisted of 4 nodes, but the possible operations between two nodes were as many as 27. Then, EAs were used to clip sub-search space from the original one and the possible operations were evaluated based on the elite model architectures in the sub-search space. After applying the EAs iteratively to generate the sub-search space and evaluating these operations that would be eliminated when their evaluations were lower than the predefined threshold, a sophisticated search space can be obtained.

IV. Discussion

ENAS aims to construct the DNNs with the higher accuracy and less human interaction. EAs are almost applied to design all kinds of networks in the existing works, including DBN, RNN, CNN, and even the generative adversarial network. Particularly, because of the intricate architecture and outstanding performance, CNN is the hot topic researched in ENAS. Therefore in Table I, the typical ENAS works related to CNN are summarized, where these methods are compared with state-of-the-art image classifiers manually designed on the common

TABLE I
COMPARISON WITH STATE-OF-THE-ART IMAGE CLASSIFIERS ON CIFAR-10, CIFAR-100 AND IMAGENET.

Architecture	**Years**	**GPU days**	**CIFAR-10(%)**	**CIFAR-100(%)**	**ImageNet(%) top-1**	**Method**
HIGHWAY [42]	2015	-	92.3	67.6	-	manual
ResNet-101 [41]	2016	-	93.4	74.8	80.1	manual
DenseNet-BC [52]	2017	-	94.8	75.6	-	manual
large-scale [4]	2017	2750	94.6	77.0	-	EA
CGP-CNN(ResNet) [36]	2017	-	94.0	-	-	EA
Hier. [37]	2017	1.5 (200 GPUs)	96.2	-	79.7	EA
NASH [44]	2017	1	94.8	76.6	-	EA
EVO-91a [53]	2018	-	94.3	71.3	-	EA
Single [22]	2019	less than 1	-	-	74.3	EA
AmoebaNet-B [26]	2019	7 (450 GPUs)	97.4	-	74.0	EA
RENASNet [40]	2019	1.5 (4 GPUs)	97.1	-	75.7	EA
AE-CNN [43]	2019	8.5	94.7	77.98	-	EA
CNN-GA [54]	2020	37.5	95.2	78.0	-	EA
CARS [28]	2020	0.4	97.4	-	75.2	EA
SI-ENAS [49]	2020	1.8	95.93	81.36	-	EA
SI-EvoNet-S [50]	2020	0.46	97.31	-	-	EA
SI-EvoNet-S [50]	2020	0.81	-	84.30	-	EA
NSENet [51]	2020	166	-	-	77.3	EA

benchmarks CIFAR-10, CIFAR-100 and ImageNet. From the table, with the development of ENAS, the automatically designed model can be obtained in a very short time with the higher performance. However, this comparison is not totally fair. The first reason is that search spaces may be different in the different works, wherein a sophisticated space even the simple random search methods can find out the model with very high accuracy [55]. The second reason is that various kinds of data augmentation techniques may be used. These data preprocess techniques can improve the model's accuracy on specific tasks. Thus, if the works with these techniques are compared with those without data augmentation, it is unfair. Therefore, in recent years, the specific benchmarks for NAS are proposed, e.g., Nas-bench-101 [56] and Nas-bench-201 [57], where the search methods are applied on the identical search space to make a fair comparison.

ENAS has achieved significant development in recent years, but the massive compute resources requirement is still an open problem. Although these speed-up strategies mentioned can reduce the time cost significantly, they often lead to suboptimal solutions. For example, reducing the search space may eliminate the optimal model architectures; the model parameter sharing strategy in the one-shot model may lead to the rank disorder phenomenon and prevent the optimal model architecture from being found out [58]. Therefore, in current situations, researchers should make a trade-off between the time cost and performance of the designed model architecture when the NAS techniques are considered.

V. CONCLUSION

This paper provides an overview of advances in ENAS. By characterizing the mathematical properties of NAS, EAs' nature advantages of this problem are highlighted. Further, these critical techniques in ENAS are reviewed along with the timeline. Specifically, EAs were applied to design the shallow forward neural networks automatically in the early time. With the renaissance of deep learning, EAs were applied to optimize DNNs hyper-parameters to improve their accuracies and gradually used to generate the entire network automatically. In recent years, ENAS mainly focuses on searching the model architectures with the higher accuracy using the less time. In summary, EAs are applied to search the neural architectures with the higher performance and enhance the degree of automatic model architecture search alternatively.

Even though ENAS has been applied to construct state-of-the-art DNNs automatically, there are still open problems, such as the unfair comparison and requirement on large computational resources. We hope this paper can provide a start point for researchers from different fields and promote the ENAS techniques in real-world applications.

REFERENCES

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

[2] E. Byla and W. Pang, "Deepswarm: Optimising convolutional neural networks using swarm intelligence," *arXiv preprint arXiv:1905.07350*, 2019.

[3] J. Liang, E. Meyerson, and R. Miikkulainen, "Evolutionary architecture search for deep multitask networks," in *Proc. ACM Genet. Evol.Comput. Conf.*, 2018, pp. 466–473.

[4] E. Real, S. Moore, A. Selle, and Saxena, "Large-scale evolution of image classifiers," in *Proc. Int.Conf. Mach. Learn.*, 2017, pp. 2902–2911.

[5] T. Elsken and J. H. Metzen, "Neural architecture search: A survey." *J. Mach. Learn. Res.*, vol. 20, no. 55, pp. 1–21, 2019.

[6] M. J. Shafiee, B. Chywl, F. Li, and A. Wong, "Fast yolo: a fast you only look once system for real-time embedded object detection in video," *arXiv preprint arXiv:1709.05943*, 2017.

[7] H. Liu, K. Simonyan, and Y. Yang, "Darts: Differentiable architecture search," *arXiv preprint arXiv:1806.09055*, 2018.

[8] B. Baker, O. Gupta, N. Naik, and R. Raskar, "Designing neural network architectures using reinforcement learning," *arXiv preprint arXiv:1611.02167*, 2016.
[9] B. Zoph and Q. V. Le, "Neural architecture search with reinforcement learning," *arXiv preprint arXiv:1611.01578*, 2016.
[10] H. Cai, T. Chen, W. Zhang, Y. Yu, and J. Wang, "Efficient architecture search by network transformation," in *Proc. 32nd AAAI Conf. Artif. Intell.*, 2018, pp. 2787–2794.
[11] H. Cai, J. Yang, and W. Zhang, "Path-level network transformation for efficient architecture search," *arXiv preprint arXiv:1806.02639*, 2018.
[12] Z. Zhong, J. Yan, W. Wu, J. Shao, and C.-L. Liu, "Practical block-wise neural network architecture generation," in *Proc. IEEE Conf. Comput.Vis. Pattern Recognit.*, 2018, pp. 2423–2432.
[13] R. Luo, F. Tian, T. Qin, and E. Chen, "Neural architecture optimization," in *Proc. Adv. Neural Inf. Process. Syst.*, 2018, pp. 7816–7827.
[14] Y. Xu, L. Xie, X. Zhang, X. Chen, G.-J. Qi, Q. Tian, and H. Xiong, "Pc-darts: Partial channel connections for memory-efficient differentiable architecture search," *arXiv preprint arXiv:1907.05737*, 2019.
[15] X. Yao, "Evolving artificial neural networks," *Proc. IEEE*, vol. 87, no. 9, pp. 1423–1447, 1999.
[16] K. O. Stanley and R. Miikkulainen, "Evolving neural networks through augmenting topologies," *Evol. Comput.*, vol. 10, no. 2, pp. 99–127, 2002.
[17] K. O. Stanley, D. B. D'Ambrosio, and J. Gauci, "A hypercube-based encoding for evolving large-scale neural networks," *Artificial life*, vol. 15, no. 2, pp. 185–212, 2009.
[18] M. Jaderberg, V. Dalibard, S. Osindero, W. M. Czarnecki, and J. Donahue, "Population based training of neural networks," *arXiv preprint arXiv:1711.09846*, 2017.
[19] A. Li, O. Spyra, S. Perel, V. Dalibard, M. Jaderberg, C. Gu, D. Budden, T. Harley, and P. Gupta, "A generalized framework for population based training," *arXiv preprint arXiv:1902.01894*, 2019.
[20] L. Xie and A. Yuille, "Genetic cnn," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2017, pp. 1379–1388.
[21] F. E. F. Junior and G. G. Yen, "Particle swarm optimization of deep neural networks architectures for image classification," *Swarm Evol. Comput.*, vol. 49, pp. 62–74, 2019.
[22] 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.
[23] H. Pham and M. Y. Guan, "Efficient neural architecture search via parameter sharing," *arXiv preprint arXiv:1802.03268*, 2018.
[24] B. Xue, M. Zhang, W. N. Browne, and X. Yao, "A survey on evolutionary computation approaches to feature selection," *IEEE Trans. Evol. Comput.*, vol. 20, no. 4, pp. 606–626, 2015.
[25] T. Shinozaki and S. Watanabe, "Structure discovery of deep neural network based on evolutionary algorithms," in *Proc. IEEE Int. Conf. Acoust. Speech Signal Process.(ICASSP)*, 2015, pp. 4979–4983.
[26] 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.
[27] T. Elsken, J. H. Metzen, and F. Hutter, "Efficient multi-objective neural architecture search via lamarckian evolution," *arXiv preprint arXiv:1804.09081*, 2018.
[28] Z. Yang, Y. Wang, X. Chen, B. Shi, C. Xu, and C. Xu, "Cars: Continuous evolution for efficient neural architecture search," in *Proc. IEEE/CVF Conf. Comput. Vit. Pattern Recognit.*, 2020, pp. 1829–1838.
[29] K. O. Stanley, "Compositional pattern producing networks: A novel abstraction of development," *Genetic programming and evolvable machines*, vol. 8, no. 2, pp. 131–162, 2007.
[30] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2012, pp. 1097–1105.
[31] C. Zhang, P. Lim, A. K. Qin, and K. C. Tan, "Multiobjective deep belief networks ensemble for remaining useful life estimation in prognostics," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 28, no. 10, pp. 2306–2318, 2016.
[32] 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 *Proc. Workshop Mach. Learn. High Perform. Comput. Environ*, 2015, p. 4.
[33] M. Gong, J. Liu, H. Li, Q. Cai, and L. Su, "A multiobjective sparse feature learning model for deep neural networks," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 26, no. 12, pp. 3263–3277, 2015.
[34] J. Liu, M. Gong, Q. Miao, X. Wang, and H. Li, "Structure learning for deep neural networks based on multiobjective optimization," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 29, no. 6, pp. 2450–2463, 2017.
[35] T. Desell, S. Clachar, J. Higgins, and B. Wild, "Evolving deep recurrent neural networks using ant colony optimization," in *Proc. Eur. Conf. Evol. Comput. Comb. Optim.*, 2015, pp. 86–98.
[36] M. Suganuma, S. Shirakawa, and T. Nagao, "A genetic programming approach to designing convolutional neural network architectures," in *Proc. Genet. Evol. Comput. Conf.*, 2017, pp. 497–504.
[37] H. Liu, K. Simonyan, O. Vinyals, C. Fernando, and K. Kavukcuoglu, "Hierarchical representations for efficient architecture search," *arXiv preprint arXiv:1711.00436*, 2017.
[38] A. Martín, R. Lara-Cabrera, and F. Fuentes-Hurtado, "Evodeep: a new evolutionary approach for automatic deep neural networks parametrisation," *J. Parallel Distrib. Comput.*, vol. 117, pp. 180–191, 2018.
[39] Y. Sun, B. Xue, M. Zhang, and G. G. Yen, "Evolving deep convolutional neural networks for image classification," *IEEE Trans. Evol. Comput.*, vol. 24, no. 2, pp. 394–407, 2019.
[40] Y. Chen, G. Meng, Q. Zhang, and S. Xiang, "Renas: Reinforced evolutionary neural architecture search," in *Proc. of the IEEE Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 4787–4796.
[41] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. of the IEEE Conf. Comput. Vis. Pattern Recognit.*, 2016, pp. 770–778.
[42] R. K. Srivastava, K. Greff, and J. Schmidhuber, "Highway networks," *arXiv preprint arXiv:1505.00387*, 2015.
[43] Y. Sun, H. Wang, B. Xue, Y. Jin, G. G. Yen, and M. Zhang, "Surrogate-assisted evolutionary deep learning using an end-to-end random forest-based performance predictor," *IEEE Trans. Evol. Comput.*, vol. 24, no. 2, pp. 350–364, 2019.
[44] T. Elsken, J.-H. Metzen, and F. Hutter, "Simple and efficient architecture search for convolutional neural networks," *arXiv preprint arXiv:1711.04528*, 2017.
[45] F. Assunção, J. Correia, R. Conceição, M. Pimenta, and B. Tomé, "Automatic design of artificial neural networks for gamma-ray detection," *arXiv preprint arXiv:1905.03532*, 2019.
[46] A. Baldominos, Y. Saez, and P. Isasi, "Evolutionary convolutional neural networks: An application to handwriting recognition," *Neurocomputing*, vol. 283, pp. 38–52, 2018.
[47] 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.
[48] G. L. F. da Silva and T. L. A. Valente, "Convolutional neural network-based pso for lung nodule false positive reduction on ct images," *Comput. Methods Progr. Biomed.*, vol. 162, pp. 109–118, 2018.
[49] 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.
[50] ——, "Efficient evolutionary search of attention convolutional networks via sampled training and node inheritance," *IEEE Trans. Evol. Comput.*, 2020.
[51] Y. Ci, C. Lin, M. Sun, B. Chen, H. Zhang, and W. Ouyang, "Evolving search space for neural architecture search," *arXiv preprint arXiv:2011.10904*, 2020.
[52] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *Proc. of the IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 4700–4708.
[53] H. Zhang, S. Kiranyaz, and M. Gabbouj, "Finding better topologies for deep convolutional neural networks by evolution," *arXiv preprint arXiv:1809.03242*, 2018.
[54] Y. Sun, B. Xue, M. Zhang, and G. G. Yen, "Automatically designing cnn architectures using the genetic algorithm for image classification," *IEEE Trans. Cybern.*, vol. 50, no. 9, pp. 3840–3854, 2020.
[55] L. Li and A. Talwalkar, "Random search and reproducibility for neural architecture search," *arXiv preprint arXiv:1902.07638*, 2019.
[56] C. Ying, A. Klein, E. Christiansen, and E. Real, "Nas-bench-101: Towards reproducible neural architecture search," in *Proc. Int. Conf. Mach. Learn.*, 2019, pp. 7105–7114.
[57] X. Dong and Y. Yang, "Nas-bench-201: Extending the scope of reproducible neural architecture search," *arXiv preprint arXiv:2001.00326*, 2020.
[58] S. You, T. Huang, M. Yang, F. Wang, C. Qian, and C. Zhang, "Greedynas: Towards fast one-shot nas with greedy supernet," in *in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2020, pp. 1999–2008.