A Cell-Based Fast Memetic Algorithm for Automated Convolutional Neural Architecture Design

Junwei Dong, Boyu Hou, Liang Feng, Huajin Tang, Kay Chen Tan, *Fellow, IEEE*, and Yew-Soon Ong, *Fellow, IEEE*

***Abstract*— Neural architecture search (NAS) has attracted much attention in recent years. It automates the neural network construction for different tasks, which is traditionally addressed manually. In the literature, evolutionary optimization (EO) has been proposed for NAS due to its strong global search capability. However, despite the success enjoyed by EO, it is worth noting that existing EO algorithms for NAS are often very computationally expensive, which makes these algorithms unpractical in reality. Keeping this in mind, in this article, we propose an efficient memetic algorithm (MA) for automated convolutional neural network (CNN) architecture search. In contrast to existing EO algorithms for CNN architecture design, a new cell-based architecture search space, and new global and local search operators are proposed for CNN architecture search. To further improve the efficiency of our proposed algorithm, we develop a one-epoch-based performance estimation strategy without any pretrained models to evaluate each found architecture on the training datasets. To investigate the performance of the proposed method, comprehensive empirical studies are conducted against 34 state-of-the-art peer algorithms, including manual algorithms, reinforcement learning (RL) algorithms, gradient-based algorithms, and evolutionary algorithms (EAs), on widely used CIFAR10 and CIFAR100 datasets. The obtained results confirmed the efficacy of the proposed approach for automated CNN architecture design.**

***Index Terms*— Cell-based space, convolutional neural network (CNN), evolutionary algorithm (EA), memetic algorithm (MA), neural architecture search (NAS).**

Manuscript received July 23, 2021; revised December 28, 2021; accepted February 14, 2022. This work was supported in part by the National Natural Science Foundation of China (NSFC) under Grant U21A20512, Grant 61876025, and Grant 61876162; in part by the Venture and Innovation Support Program for Chongqing Overseas Returnees under Grant cx2018044 and Grant cx2019020; in part by the Research Grants Council of the Hong Kong SAR under Grant PolyU11202418, Grant PolyU11211521, and Grant PolyU11209219; and in part by the Centre for Frontier AI Research, A*STAR. *(Corresponding author: Liang Feng.)*

Junwei Dong, Boyu Hou, and Liang Feng are with the College of Computer Science, Chongqing University, Chongqing 400044, China (e-mail: jwdong@cqu.edu.cn; byhou@cqu.edu.cn; liangf@cqu.edu.cn).

Huajin Tang is with the College of Computer Science and Technology, Zhejiang University, Hangzhou 310027, China, and also with the Zhejiang Laboratory, Hangzhou 311121, China (e-mail: htang@ziu.edu.cn).

Kay Chen Tan is with the Department of Computing, The Hong Kong Polytechnic University, Hong Kong (e-mail: kctan@polyu.edu.hk).

Yew-Soon Ong is with the Center for Frontier AI Research, Agency for Science, Technology and Research (A*STAR), Singapore 138632, and also with the School of Computer Science and Engineering, Nanyang Technological University (NTU), Singapore 639798 (e-mail: asysong@ntu.edu.sg).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TNNLS.2022.3155230.

Digital Object Identifier 10.1109/TNNLS.2022.3155230

I. Introduction

CONVOLUTIONAL neural networks (CNNs) [1]–[5] have attained great success in various tasks in the field of computer vision. However, traditional CNN architectures are often designed manually by human experts with domain-specific knowledge, which is a time-consuming and error-prone process. Therefore, in recent years, to facilitate and automate the architecture design of the deep neural network, neural architecture search (NAS) has been proposed in the literature, which has attracted growing interest from both industry and academia. NAS has obtained superior performance in terms of both design efficiency and architecture quality in contrast to human-designed models on a variety of learning tasks, such as object detection [6], [7], semantic segmentation [8]–[10], and natural language processing [11], [12].

According to a recent survey [13], existing NAS algorithms can be categorized into three classes: reinforcement learning (RL)-based algorithms, gradient-based algorithms, and evolutionary search (ES)-based algorithms. In particular, examples of RL-based algorithms include that presented a study on training recurrent networks by RL to maximize the expected accuracy of the generated architectures on a validation set and [14] that used RL to train a controller to generate specific domain language for updating the predefined NAS optimizer. However, despite the success enjoyed by RL-based methods, it is worth noting here that these algorithms require a large number of computing resources and are often computationally expensive. Next, in contrast to RL-based methods, gradient-based algorithms are much more computationally efficient. For instance, Liu *et al.* [15] formulated the problem of NAS in a differentiable manner (i.e., DARTS) and conducted continuous relaxation of the architecture representation that allows efficient search of the architecture using gradient descent. Furthermore, Xu *et al.* [16] presented an extension of DARTS by sampling a small part of the supernet to reduce the redundancy in exploring the network space, thereby performing a more efficient search without comprising the performance. Although GD algorithms are efficient in computing, these methods usually suffer from large memory and inappropriate relaxation in applying the gradient-based optimization approach. Moreover, as most exiting gradient-based algorithms depend on an auxiliary HyperNet, these methods also highly rely on domain-specific knowledge in HyperNet construction.

In contrast to RL- and gradient-based algorithms, the ES-based algorithm is a type of nature-inspired population-based optimization approach, which has strong global search capability and has no requirements, such as convex nature, derivability, and explicit mathematic formulation, on the objective functions. Over the years, quite a number of ES-based algorithms have been proposed for NAS. In particular, one of the earliest ES-based methods is NeuroEvolution of Augmenting Topologies (NEAT) proposed by Stanley and Miikkulainen [17], which optimizes the topology structure and weights of a neural network simultaneously. Next, Real *et al.* [18] employed simple evolutionary techniques to discover neural network architectures for the CIFAR-10 and CIFAR-100 datasets. Xie and Yuille [19] used a genetic algorithm with an encoding method representing each network structure in a fixed-length binary string to learn deep CNN structures automatically. To further improve the NAS performance, Real *et al.* modified the tournament selection evolutionary algorithm (EA) by introducing an age property to favor the younger genotypes, which obtained superior performance over handcrafted models [20]. Sun *et al.* [21] proposed a variable-length encoding technique and a novel fitness evaluation method in the genetic algorithm to explore the architecture space efficiently. More recently, Yang *et al.* [22] developed an efficient continuous evolutionary approach for searching neural networks, which directly inherits both the SuperNet and the population over generations. Zhang *et al.* proposed a computationally efficient ES of convolutional networks based on a directed acyclic graph (DAG), which randomly samples and trains parents on each minibatch of the training data. However, despite the attractive global optimization capability of ES, it is noted that ES requires a large number of performance evaluations in its iterative search process, while the training of deep neural networks is itself a computationally expensive task.

Keeping the above in mind, in this article, we propose an efficient memetic algorithm (MA) for automated convolutional neural architecture design, which takes both high-quality CNN architecture search capability and search efficiency into consideration. In particular, as the processes of convolution and pooling in CNN correspond to the learning of deep features and the reduction of the dimensionality of the convolution layer output, respectively, these two processes serve as the global learning and local refinement in the deep learning procedure. Taking this cue, based on the popular cell-based search space [23], we first propose a new cell-based search space that considers either convolution or pooling between hidden layers and possesses a smaller search space in contrast to existing cell-based search spaces. Next, with the new designed search space, we propose an efficient MA with global search exploring the architectures of convolution, while local search exploits the operations of pooling. Moreover, to further enhance the search efficiency of CNN architectures, as our proposed algorithm separates the searches of convolution and pooling architectures, which could lead to CNN architectures with stable performance, we propose to use a one-epoch-based evaluation strategy that estimates the performance of the obtained CNN model by training it on the target dataset for only one epoch. Last but not least, to validate the efficacy of our proposed approach for NAS, comprehensive empirical studies are conducted on the commonly used datasets, i.e., CIFAR-10 and CIFAR-100 benchmarks, against 34 existing state-of-the-art algorithms for NAS, including manual algorithms [3], [5], [24]–[28], RL-based algorithms [23], [29]–[32], gradient-based algorithms [15], [16], [33]–[39], EA-based algorithms [18], [20], [22], [40]–[45], and so on [46], [47].

To summarize, the main contributions of this article are given as follows.

1) In order to improve the search efficiency and effectiveness of CNN architectures, we propose a new and separated search space based on cell-based space and design a new encoding strategy to represent the CNN architecture. Due to the restrictions and splitting of the search space, the corresponding optimization search can be performed efficiently and effectively. Moreover, the improved encoding strategy allows the complete representation of structural information and, thus, facilitates the combination of operators during the search procedure.
2) To search for high-quality architectures, we design a fast MA for NAS with global and local search in the separated search space, respectively. Due to the separation of convolution and pooling operators in contrast to the existing joint search, our proposed method can discover well-performed architectures efficiently.
3) For further accelerating the NAS process, a one-epoch-based evaluation technique is proposed for model performance evaluation, which significantly reduces the computational overhead in the evaluation of the found CNN architectures.

The rest of this article is organized as follows. A brief review of NAS and MA is introduced in Section II. Section III gives the details of our proposed algorithm for efficient CNN architecture design. Next, the experimental setup and empirical results obtained by our proposed approach over 34 existing state-of-the-art NAS algorithms on CIFAR-10 and CIFAR-100 are presented and discussed in Section IV. Finally, the conclusive remarks of this article and the corresponding future work are summarized in Section V.

II. Preliminary

In this section, we first discuss the three key design issues of NAS for CNN architecture design. Next, a brief introduction to MA is also presented in this section.

A. NAS for CNN Architecture Design

NAS denotes the process of automating the architecture engineering of neural networks, which is one of the main research directions in automated machine learning (AutoML) [48]. As depicted in Fig. 1, NAS mainly involves three key design issues: search space, search strategy, and performance estimation strategy [48]. In the literature, most of the existing NAS algorithms have been proposed for searching optimal CNN architectures. In particular, there are typically two types of search spaces for CNN architecture search.

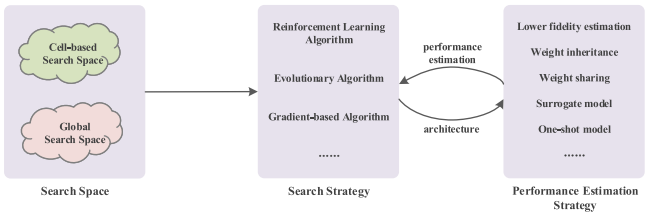


Fig. 1. Three key design issues, i.e., search space, search strategy, and performance estimation, in NAS algorithm design.

The first type is global search space that represents the entire neural architecture of CNN by graphs [19], [31], [49]–[52]. This type of search space usually allows large degrees of freedom regarding the arrangement of operations. Furthermore, the other type is cell-based search space, which stacks smaller sized graphs to form a larger CNN architecture [23], [41], [53], [54]. The repeating graphs or structures are referred to as cells. The design of cell-based search space is based on the observation that many effective and high-quality handcrafted architectures are designed with repetitions of fixed structures. Today, toward efficient NAS, cell-based search space is more preferred in the design of CNN architecture search algorithms since it possesses a smaller search space in contrast to the global search space.

Next, with a well-designed CNN search space, the search strategy then kicks in to explore the architecture space with the aim of finding high-quality CNN structures efficiently. As discussed in Section I, existing search strategies can be categorized into three classes: the RL-based algorithm [29], [31], [55], [56], the gradient-based algorithm [15], [16], [30], [57], [58], and the ES-based algorithm [18]–[20], [22], [40]. To date, because of the nonconvex and complex nature of the problem of CNN architecture search, ES-based methods have attracted increasing attention in the literature due to their strong optimization capability and easy implementation. However, as EA contains an iterative search process, the design of an efficient ES approach is a nontrivial task to enable automated CNN architecture search in practice.

Last but not least, the third key design issue is the performance estimation strategy, which provides evaluations on the quality of the found neural architecture, while the NAS process progresses online. In the literature, early attempts often train the found neural architectures on a complete training dataset until convergence, which requires a large amount of computational cost. Moreover, to accelerate the evaluation process, a number of performance estimation strategies have been proposed to evaluate the found neural architecture quickly, including lower fidelity estimation [23], [59], [60], weight inheritance [42], [61], weight sharing [29], [34], the surrogate model [53], [62], and the one-shot model [15], [16], [57]. More literature reviews on NAS can be referred to [13] and [49].

B. Memetic Algorithm

MA has materialized as a form of population-based global search with lifetime learning as a separate process capable of local refinement for accelerating search [63]–[68]. In the literature, many dedicated MAs have been crafted to solve domain-specific problems more efficiently, including permutation flow shop scheduling [69], quadratic assignment problem [70], capacitated arc routing problem [71], and feature selection [72]. These studies on MAs have demonstrated that they converge to high-quality solutions more efficiently than their conventional counterparts in solving complex optimization problems.

In this article, we propose an efficient MA for automated CNN architecture search, which contains the global search of convolution operation structures and local search of pooling operation structures. In this way, the proposed method is able to explore the CNN architecture space effectively and efficiently by searching subspaces of convolution and pooling structures, respectively. The details of our proposed algorithm, including search space, search operators, and neural architecture evaluation, are presented in Section III.

III. Proposed Algorithm for Automated CNN Architecture Design

This section presents the details of our proposed Memetic Search of Neural Architecture Search for automated CNN design (MSNAS for short). In particular, the outline of the proposed algorithm is summarized in Fig. 2. As depicted in Fig. 2, the search process starts with search space construction and population initialization; after evaluation of the generated CNN architectures, the ES process will be performed iteratively until a predefined stopping criterion is satisfied. The obtained best CNN architecture is then the output of the proposed algorithm. In contrast to existing ES-based NAS algorithms, the proposed algorithm mainly differs in the following aspects: search space and solution encoding, search strategy, and architecture performance evaluation. In particular, the proposed search space contains two subspaces: one is for convolution operators and the other is for pooling operators. Based on this, a modified DAG [19] is developed for solution encoding. Next, the proposed memetic search also contains two types of search operators, i.e., the global search for convolution architecture and local refinement for pooling architecture. For architecture performance evaluation, a new and efficient performance evaluation strategy, which trains the optimized CNN models for only one epoch, is developed to roughly estimate the performance of different found CNN architectures.

In what follows, the details of the proposed search space and solution encoding, search strategy, and architecture performance evaluation are presented.

A. Search Space

The search space for NAS can be treated as a huge computational graph, and the objective of NAS is to find the subgraph in this space with a branch of nodes $\mathbf{E}$, which performs well on a given learning task, such as image classification, pattern recognition, and speech recognition [13], [48]. In the found subgraph, each node $E^{(i)}$ is a hidden feature map that receives the flow of information processed by specific operators from its parent nodes. In the context of CNN, as aforementioned,

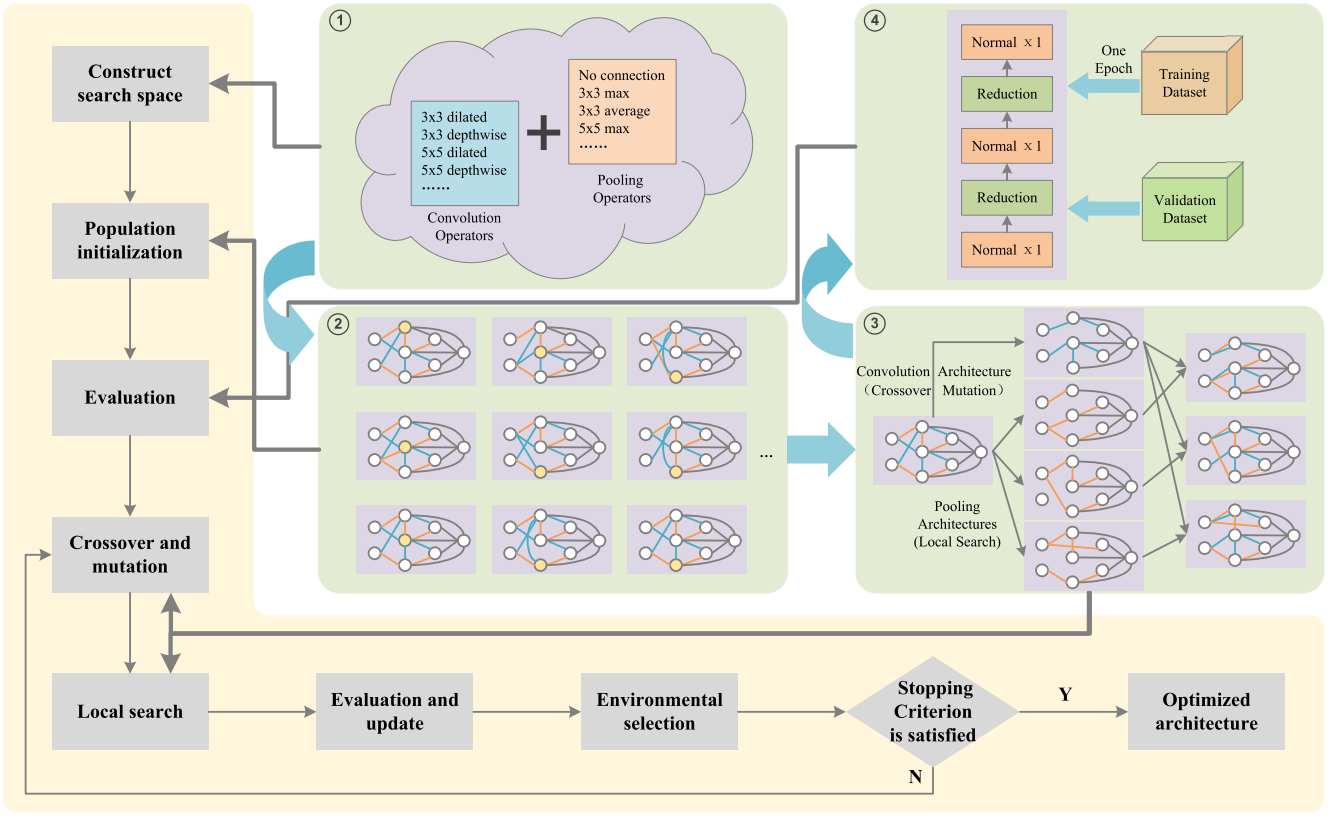


Fig. 2. Outline of our proposed MSNAS for automated CNN architecture design. ① + ②: search space and solution encoding; ③: search strategy; and ④: architecture performance evaluation.

the cell-based search space has achieved significant successes in automated CNN designs [13], [23]. In the cell-based search space [23], a complete network is a stack of normal cells and reduction cells (each cell receives the output of its previous two cells), which are small graphs containing a series of neural nodes, where each one receives two information flows that are processed by different operators. Moreover, the operators for processing information flow between neural nodes include pooling and convolution filters with different filter sizes, filter numbers, sliding windows, and so on. Each node receives information processed by the filters from two of its parent nodes, and the computation flow for a node can be defined by

$$E^{(\text{node}_i)} = o^{(i1)}\left(\left\{E^{i1}\right\}\right) + o^{(i2)}\left(\left\{E^{i2}\right\}\right) \quad (1)$$

where $o^{(\cdot)}$ denotes a particular convolution or pooling filter. As illustrated in Fig. 3, in existing cell-based search space, for each of the two parent neural nodes, the convolution or pooling operator is randomly selected to process the information for the child neural node. However, as aforementioned, convolution and pooling focus on different aspects in feature learning, i.e., convolution concentrates on the learning of deep features, while pooling is more on the dimensionality of the learned features. Taking this cue, instead of randomly select convolution and pooling as the process operator, here, we propose to separate these two operators and specify each for one parent neural node, respectively [see Fig. 3 (right)]. This cannot only reduce the volume of the search space

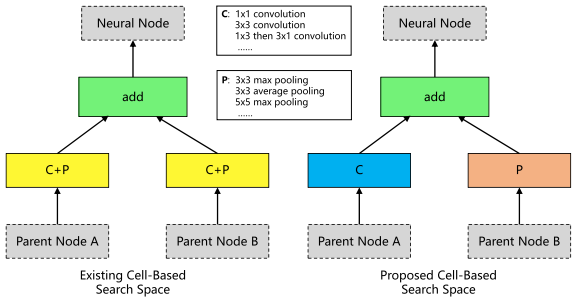


Fig. 3. Illustration of computation flows in existing and the proposed cell-based search space. A rectangle with a solid line denotes an operation, which can be either a convolution or a pooling operator, and different colors mean different types of operations. The gray rectangle denotes the neural node, which is also known as a feature map. Arrows in the figure represent the direction of information flow. Left: existing cell-based space with randomly processed information flow. Right: our proposed cell-based space with different types of information flow.

but also improve the stability of the search for CNN architectures. In addition, the dimension of the data in a cell is always kept unchanged, and a "FactorizedReduce" block, which is commonly used in cell-based NAS algorithms [15], [16], [23], [73], is adopted to halve the feature map size and double the number of channels before each reduction cell.

1) Proposed Cell-Based Search Space and Encoding Scheme: As illustrated in Fig. 4, the proposed cell-based search space divides the original cell-based search space

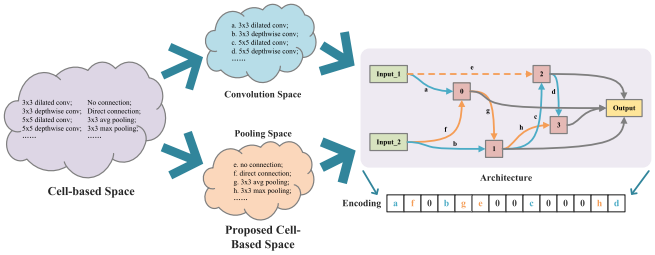


Fig. 4. Search space and encoding of the proposed MSNAS. The global space and the local space in the modified space are subspaces of the cell-based space. In the architecture, the blue solid lines and the yellow solid lines are from the global space and the local space, respectively, and the yellow dashed lines mean that there is no connection between the two corresponding nodes. In the code, the letters with different colors declare the disparate types of operators from these two spaces.

into two subspaces, which are the convolution space and the pooling space that contain various convolution operators and different pooling operators, respectively. Therefore, each neural node in the found CNN receives two flows of information from its parent nodes: one is operated by convolution (see the blue line in Fig. 4), and the other is operated by pooling (see the orange line in Fig. 4), which can be represented by

$$E^{(\text{node}_i)} = o^{(i1)}_{\text{convolution}}\left(\{E^{i1}\}\right) + o^{(i2)}_{\text{pooling}}\left(\{E^{i2}\}\right) \tag{2}$$

where $o^{(i1)}_{\text{convolution}}$ denotes a convolution operator selected from the convolution space, while $o^{(i2)}_{\text{pooling}}$ gives a pooling operator found in the pooling space.

Moreover, based on the DAG in [19], we propose a modified DAG to represent a found CNN architecture. In particular, an example of the proposed encoding scheme is given in Fig. 4. As illustrated in Fig. 4, there are four neural nodes denoted by "1," "2," "3," and "4" in the DAG. Neural node "0" receives information flow from "Input 1" and "Input 2" using convolution "a" and pooling "f," respectively. For neural node "1," it has three possible information flows, which are neural node "0," "Input 1," and "Input 2." In Fig. 4, it receives information flows from "Input 2" and neural node "0" via convolution "b" and pooling "g," respectively. Accordingly, we can see that neural nodes "2" and "3" will have four and five possible information flows, and these two nodes receive information from the pair of ("Input 1" and neural node "1") and (neural node "1" and neural node "2"), respectively, in Fig. 4.

Next, for solution encoding, in contrast to the "01" string encoding commonly used in the literature [19], which can only represent the connections between neural nodes, we consider both alphabet and "0" strings to describe the operator and connection information, respectively. Similar to [19], a string is employed to indicate the information flows for a cell. In particular, each node i owns $i + 2$ bits where the first two bits denote the connections between node i and the two input nodes, and the remaining bits indicate the relationship between node i and the other nodes that have smaller index values than node i. Moreover, the alphabets are utilized to represent the different types of operators in our encoding strategy, e.g., "a" denotes the convolution operator "DilConv 3×3" (dilated convolution with 3×3 kernel size) and "g" denotes the pooling operator "AvgPool 3×3" (average pooling with 3×3 sliding window). As depicted in Fig. 4, the example of a cell architecture represented by the proposed DAG can be encoded as "af0bge00c000hd." Each node receives only two information flows that are processed by the convolution and pooling operation (labeled by the alphabets in the solution), respectively. "0" means that there is no connection between the current node and the input flow represented by the corresponding bit. For example, in the solution, the first two bits "af" denotes the information flows for neural node "0," while the next three bits "0bg" represent the three possible information flows for neural node "1." Furthermore, the bits "e00c" denote that from the four possible information flows (i.e., "Input 1," "Input 2," neural node "0," and neural node "1"), and "Input 1" and neural node "1" will be received by neural node "2" via pooling "c" and convolution "e, " respectively. The last five bits then represent the connection and operation information for neural node "3," which takes the information flows of neural nodes "1" and "2" as inputs via pooling "h" and convolution "d," respectively.

2) Population Initialization: Based on the proposed search space and encoding scheme, in this section, we introduce the details of population initialization in our proposed memetic search for automated CNN architecture design. In particular, as summarized in Algorithm 1, for M number of hidden neural nodes, we first calculate the total number of possible information flows of all the hidden neural nodes, which is $(((3 + M) * M)/2)$.[1] Next, for each solution, we randomly assign a particular convolution operation $o_{\text{convolution}}$ from the convolution search space O_{con} for a randomly selected information flow for each hidden neural node (lines 4 and 5 in Algorithm 1). However, to increase the diversity of the initialization of convolution operation, we will reassign the convolution to another selected information flow for each hidden neural node if the number of generated solutions are less than $(((3 + M) * M)/2)$. In this way, all the possible information flows for each hidden neural node can be assigned with a convolution operation (lines 6–8 in Algorithm 1). Moreover, the pooling operation will be randomly selected from the pooling search space O_{pool} and assigned to an unassigned information flow of each hidden neural node randomly (line 9 in Algorithm 1). The process of assigning convolution and pooling operations will be performed repeated if the predefined numbers of solutions are generated.

B. Search Strategy

In this section, the details of our proposed strategy, including global search in convolution search space and local refinement in pooling space, are given. Referring to [74]–[79], we also first design global search operators that consist of crossover and mutation to pay attention to the information

[1]As discussed in Section III-A1, the first hidden neural node has two possible information flows, and the second node then has three possible information flows. By this analogy, the third and fourth hidden neural nodes have four and five possible information flows, respectively, and the total number of possible information flows is $2 + 3 + 4 + 5 + \cdots + (M + 1)$, which is equal to $(((3 + M) * M)/2)$.

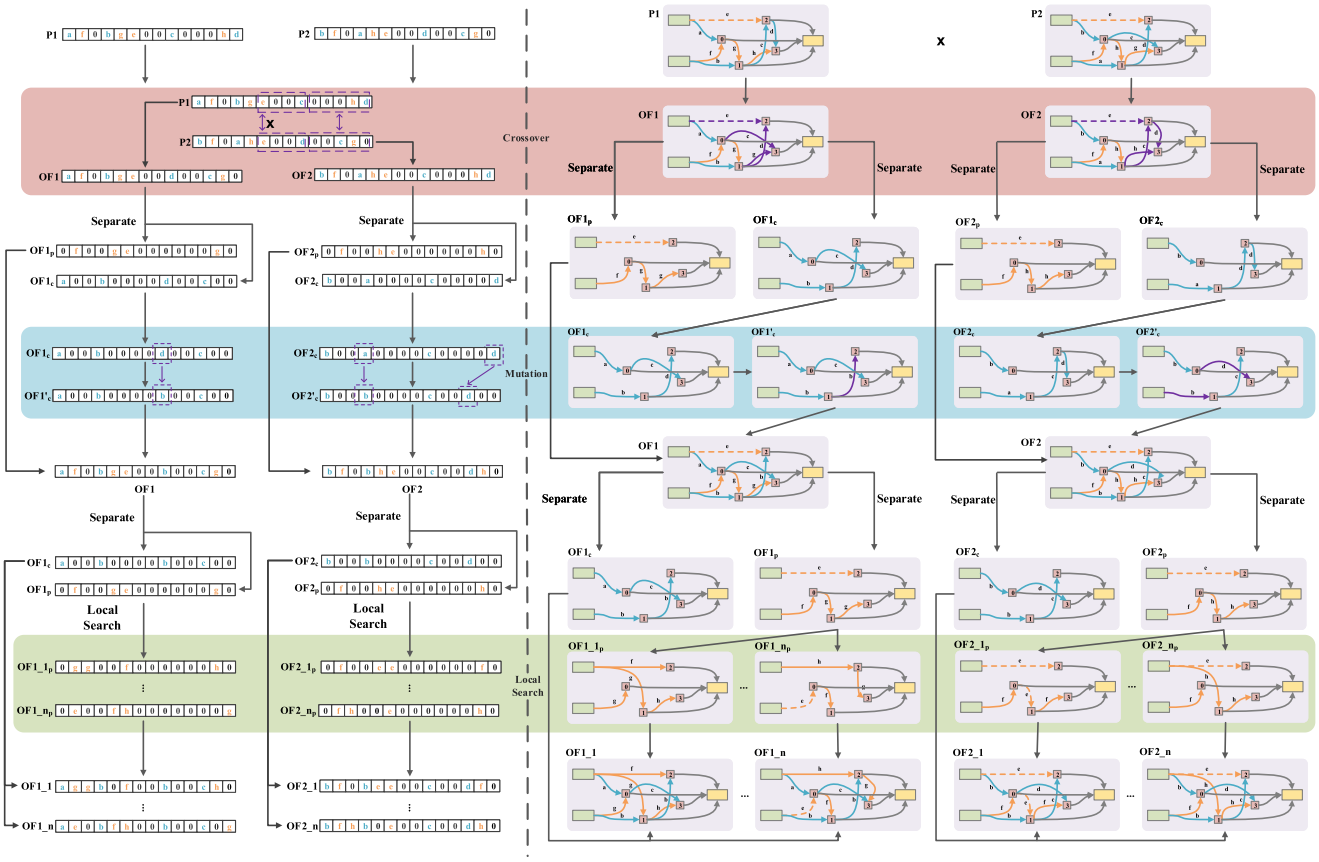


Fig. 5. Illustration of offspring generation by our proposed search strategy, i.e., global search including crossover and mutation, and local search. (For two selected parent solutions P1 and P2, two offspring solutions OF1 and OF2 will be generated. The DAGs on the right side of Fig. 5 are the graphical illustration of the corresponding proposed search operators given on the left-hand side. In the figures, blue symbols and arrows denote convolution operations, while orange symbols and arrows give pooling operations.)

exchange between individuals in the population, and then, the local search operator is devised to focus on the pooling information optimization for each individual. First, as illustrated in Fig. 5, two parent solutions are selected via roulette wheel selection [80] from the population, i.e., P1 and P2. Next, crossover will be performed to generate the offspring solutions OF1 and OF2 based on P1 and P2. Furthermore, for each offspring solution, we generate two strings, e.g., $OF1_c$ and $OF1_p$, by separating the convolution and pooling operations in the solution, respectively. The mutation will then be conducted probabilistically with the separated convolution strings of the offspring solutions, i.e., $OF1_c$ and $OF2_c$, to explore the convolution search space and generate two new convolution strings, i.e., $OF1_c$ and $OF2_c$. On the other hand, local search will also be performed probabilistically on the pooling strings, i.e., $OF1_p$ and $OF2_p$, to exploit the pooling structures. The resultant new pooling strings i.e., $OF1_p$ and $OF2_p$, will be combined with the convolution strings, i.e., $OF1_c$ and $OF2_c$ to give the offspring solutions OF1 and OF2.

In what follows, the details of our proposed global and local search are presented.

1) Global Search: As discussed above, the global search has two phases, i.e., crossover and mutation. Considering the encoding of a cell here is essentially a string encoding, we propose a variant of the multipoint crossover [81] to

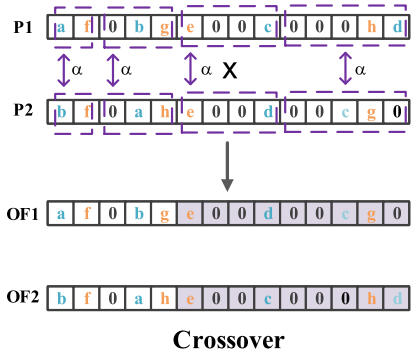


Fig. 6. Detailed description of our proposed crossover operator.

exchange hidden neural nodes of the parent solutions. In particular, for two parent solutions P1 and P2, the index points for crossover are first identified according to the number of possible information flows for each hidden neural node (as discussed in Section III-A1). As can be observed in Fig. 6, there are three index points have been identified, and each segment in the string represents a hidden neural node. Next, each segment (or hidden neural node) will be exchanged under a predefined probability α to generate two new convolution strings OF1 and OF2.

Algorithm 1 Population Initialization

Input: population size S, number of hidden neural nodes M, the search space O
Output: the initial population P_0

$P_0 \leftarrow \phi$;
$S_c \leftarrow$ total number of possible information flows for all the hidden neural nodes $\frac{(3+M)\times M}{2}$;
$s \leftarrow 0$;
while s < S **do**
  $ind \leftarrow$ initialize an individual via randomly assigning a specific convolution operator $o_{convolution} \in O_{con}$ for a randomly selected information flow for each hidden neural node;
  if $s < S_c$ **then**
    Reassign the convolutional operators for ind to make sure that each possible information flow of all the hidden neural nodes has been assigned at least one convolutional operator;
  end
  Randomly assign a pooling operator $o_{pooling} \in O_{pool}$ for each hidden neural node in ind ;
  $P_0 \leftarrow P_0 \cup ind$;
  s $\leftarrow$ s + 1 ;
end

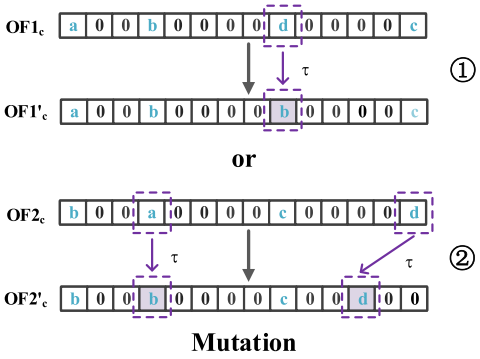


Fig. 7. Detailed description of the proposed mutation operator.

After crossover, the generated two offspring solutions will undergo mutation to further explore the convolution search space. As illustrated in Fig. 5, before mutation, each of the offspring solutions is first to be separated into two strings, i.e., convolution string and pooling string, and only the convolution string of the offspring solution will undergo mutation. In particular, the mutation is performed with respect to each hidden neural node under a predefined probability τ. As illustrated in Fig. 7, we propose two mutation operators that are randomly selected to execute when the mutation is triggered. As can be observed in Fig. 7, the first mutation operator focuses on the change of convolution operators and kept the connection of information flow unchanged. While the other mutation operator keeps the selected convolution operator and tries to change the information flow to another unconnected one of the corresponding hidden neural nodes (i.e., indicated by "0" in the dimension representing this hidden neural node in the string).

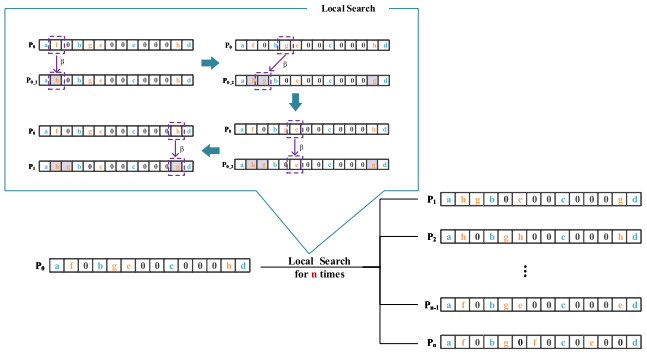


Fig. 8. Illustration of the proposed local search.

2) Local Search: Local search is to further perform individual learning on the generated offspring solutions via exploiting the pooling space. In particular, the pseudocode and an illustration of the proposed local search are given in Algorithm 2 and Fig. 8, respectively. As can be observed, the proposed local search is performed with respect to each hidden neural node under a predefined probability β in the pooling string of a given individual P. There are also two types of search in the proposed local search operator: change the pooling operator and keep the information flow for pooling unchanged, while the other one preserves the pooling operator but change the information flow to another unconnected input of the node (i.e., indicated by "0" in the dimension representing this hidden neural node in the string). These two types of search are randomly selected when performing the local search process. Furthermore, in Algorithm 2, the parameter N is to control the times of performing a local search on a particular individual. It is straightforward to see that N balances the tradeoff between local exploitation and search efficiency of the proposed memetic search for automated CNN architecture design. Last but not least, the best individual found via local search will survive and be used to replace individual P in the population.

C. Performance Estimation Strategy

As the performance evaluation of a particular neural network architecture requires the training of the corresponding neural weights, the fitness evaluation of individuals along the evolutionary CNN architecture search process is often extremely computationally expensive. To alleviate the computational burden of performance evaluation for CNN architecture search, a number of performance estimation approaches have been proposed in the literature [15], [16], [23], [53], [57], [60]–[62]. However, these approaches may either still not be computationally efficient or rely on particular neural network architectures. Recently, Zheng *et al.* [73] showed that the evaluation of different neural architectures in the early stage of training can be used as the performance estimation in comparing the architectures. Based on this, in this article, we propose to evaluate the performance of the found CNN architecture according to the first epoch of its training on a

Algorithm 2 Proposed Local Search Strategy

Input: individual P, iteration N
Output: the best found individual P_{best}

1 $P_{temp} \leftarrow \phi$;
2 $P_{best} \leftarrow P$;
3 $P_p \leftarrow$ the pooling string of P ;
4 $P_c \leftarrow$ the convolution string of P ;
5 $n \leftarrow 0$;
6 **while** $n < N$ **do**
7 $P_{temp} \leftarrow$ change the pooling operator or change the information flow for pooling of each hidden neural node of P_p probabilistically;
8 $P_{temp} \leftarrow$ reconstruct a new individual with the P_{temp} and P_c ;
9 **if** P_{temp} is fitter than P_{best} **then**
10 $P_{best} \leftarrow P_{temp}$
11 **end**
12 $n \leftarrow n + 1$;
13 **end**

given dataset. Moreover, as existing CNN architectures are often based on the stack of normal cell and reduction cell [15], [48], [50], the performance of the whole CNN architecture could be estimated by a simplified combination of normal and reduction cells [23]. Therefore, to further reduce the computational cost for architecture evaluation, we propose to train a simplified combination of the found normal and reduction cell for evaluation, which is "NRNRN," where "N" and "R" stand for the normal cell and the reduction cell, respectively.

D. Summary of the Proposed Algorithm

The pseudocode of our proposed memetic search for automated CNN architecture design is summarized in Algorithm 3. In particular, first, we divide the cell-based search space into two subspaces, i.e., convolution space and pooling space, where one only contains convolution operators with different filters, and the other only has pooling operators possessing diverse sliding windows (line 1). Next, the S number of individuals will be initialized using the population initialization approach introduced in Algorithm 1 (line 2). For each individual, it contains two cells: one is for the normal cell, and the other is for the reduction cell. According to [13], the number of hidden neural nodes M for both normal cell and reduction cell is set as 4. After evaluation of the generated population using the proposed performance estimation approach discussed in Section III-C, ES for CNN architecture optimization, including global crossover and mutation, and local individual reinforcement is performed iteratively, until the predefined stopping criteria are satisfied (lines 3–11). For the selection process (line 10), considering both individual quality and diversity, both elitism selection [81] and tournament selection [82] are employed. In particular, for selecting an individual for survival in the next generation, the first half of the population is selected via elitism selection, and the other half is selected by tournament selection. Finally, the output of the proposed memetic search is then the best individual, containing the found best normal and reduction cell. The corresponding CNN architecture will be obtained via a stack of the optimized normal and reduction cells. Particularly, according to [13], [16], [24], and [74], in this study, the CNN architecture is achieved by stacking the normal cell structure for six times, which is a common scheme $(N \times 6)R(N \times 6)R(N \times 6)$, where "N" and "R" stand for normal and reduction cells, respectively.

Algorithm 3 Pseudocode of the Proposed MSNAS

Input: population size S, the set of convolution operators O_{con}, the set of pooling operators O_{pool}, the training dataset D_{train} and validation dataset D_{val}
Output: the best CNN architecture

1 $O \leftarrow$ Construct the search space using O_{con} and O_{pool};
2 $P_0 \leftarrow$ Initialize a population of size S, using the proposed initialization approach in Alg. 1 based on O;
3 Train and evaluate the generated individuals in P_0 on D_{train} and D_{valid}, respectively, using our proposed evaluation strategy;
4 $t \leftarrow 0$;
5 **while** the termination condition is not satisfied **do**
6 $PS \leftarrow$ Select two parent individuals using roulette wheel selection from P_t;
7 $Q_t \leftarrow$ Generate two new offsprings by the proposed crossover and mutation operators using PS;
8 $Q_t \leftarrow$ Produce additional individuals with a size of N for each individual in Q_t with the local search strategy;
9 Train and evaluate the individuals in Q_t on D_{train} and D_{valid}, respectively;
10 $P_{t+1} \leftarrow$ Select S different architectures from $P_t \cup Q_t$ via elitism and tournament selection strategies;
11 $t \leftarrow t + 1$;
12 **end**
13 Select the best individual from P_t and decode it into the corresponding convolutional neural network.

IV. Empirical Study

In this section, comprehensive empirical studies are conducted to validate the performance of our proposed memetic search for automated CNN architecture design. In particular, we first give the experimental configurations, including benchmark datasets, baseline algorithms, and parameter settings. Next, the comparison results in terms of the test error, the training time, and the number of parameters are presented and discussed.

A. Benchmark Datasets

Two commonly used benchmark datasets, i.e., CIFAR10 [83] and CIFAR100 [83], are considered in the empirical comparison to investigate the performance of our proposed algorithm. In particular, CIFAR 10 has ten classes, including 50 000 training images and 10 000 test images. Each class has 6000 images, and the dimension

of each image is 32 × 32. Furthermore, in contrast to CIFAR 10, CIFAR 100 is a more complex dataset that includes 100 classes with 600 images per class. There are 500 training images and 100 testing images in each class. For more details of these two datasets, interested readers can refer to [83].

Moreover, in this study, the search of CNN architecture on the CIFAR 10 and CIFAR 100 dataset is performed separately. The performance evaluation of the transferred CNN architecture that is obtained on CIFAR 10 and CIFAR 100 datasets is also conducted.

B. Baseline Algorithm

For baseline algorithms, 34 number of state-of-the-art NAS methods are compared. In particular, according to a recent survey [13], the compared algorithms are divided into three categories as follows.

1) The first category includes state-of-the-art CNNs which are designed by human expert. Particular algorithms contain VGG [24], SENet [25], ResNet [3], Wide ResNet [26], ResNetXt-29 [27], DenseNet [5], and ShuffleNet [84].
2) The second category contains a number of recently proposed nonevolutionary NAS algorithms, including both RL-based methods and gradient-based (GD) based methods. In particular, the compared algorithms include NASNet [23], MetaQNN [45], ENAS [29], ProxylessNAS [30], PNASNet [53], MdeNAS [73], DARTs [15], P-DARTS [33], SNAS [34], Firefly [35], and AdaptNAS-S [36].
3) The third category includes various state-of-the-art evolutionary NAS approaches for CNN architecture design. The compared particular algorithms are large-scale evolution [18], hierarchical evolution [41], AmoebaNet [20], LEMONADE evolution [42], NSGANet [43], AE-CNN [44], CARS [22], and SI-EvoNet-S [40].

Moreover, for the compared algorithms, to ensure that there is no performance loss by algorithm reimplementation, the best results of these algorithms published in the corresponding papers are directly used here for comparison.

C. Experimental Configuration

For experimental configuration, according to [45], the evolutionary parameter settings are summarized in Table I. In Table I, the parameters α, τ, and β denote the crossover probability, mutation probability, and local search probability, which are discussed in Section III-B for controlling each hidden neural node to undergo node exchange, convolution mutation, and pooling local search, respectively. To alleviate the computational cost for ES, each individual performs a local search three times in all the experiments conducted in this study.

Moreover, according to [13], we use half of the CIFAR10 training set as the validation set (training set proportion is set to 0.5) and reduce the image size by half (image compression rate is set to 0.5). Similar to [16] and [74], each cell contains four nodes (the number of neural nodes is set to 4), and the number of initial convolution channels is 16, which are common settings for NAS in the literature. In particular, there are two sets of parameter settings, one is for the ES process ("Evolution" column), and the other is for the evaluation of the found CNN architectures ("Evaluation" column). To optimize the network weight for evaluation, the common stochastic gradient descend with momentum (SGDM) [16], [73] is employed. The corresponding configurations, including initial learning rate and weight decay, are given in Table I as well.

TABLE I
HYPERPARAMETER SETTINGS IN MSNAS

Parameters	Evolution	Evaluation
Population size	20	-
Generations	20	-
Crossover probability	0.5	-
Mutation probability	0.5	-
Local search probability	0.5	-
Local search iteration	3	-
Image compression rate	0.5	1
Training set proportion	0.5	1
Number of neural nodes	4	-
Initial convolution channels	16	32
Batch size	64	96
Epoch	1	600
Optimizer	SGDM	SGDM
Initial learning rate	0.025	0.025
Weight decay	3e-4	3e-4
Learning rate schedule	Cosine Annealing	Cosine Annealing

Finally, in this study, we conduct two sets of empirical studies. One is the comparison against state-of-the-art NAS approaches for automated CNN architecture design, while the other is to investigate the effectiveness of the proposed search space and the simplified structure used in the performance evaluation, as introduced in Section III-C. All the experiments conducted are based on the Nvidia GeForce RTX 2080Ti.

D. Results and Discussion

In this section, the results obtained by our proposed algorithm over ten independent runs and the compared algorithms are presented and analyzed. In particular, Tables II and III summarized the obtained results in terms of the best test error, the average test error, the number of parameters and consumed computational cost, and the optimization method used on CIFAR10 (C10) and CIFAR100 (C100) datasets, respectively. In particular, for computational cost, referring to existing studies, the "GPU Days" are considered as the quantitative metric, which is calculated via multiplying the used number of GPU cards and the running time cost by the algorithm in a day [13]. For instance, we used four GPU cards in our experiments, and the running time of the proposed MSNAS is 1.38 h on the CIFAR10 dataset. The corresponding GPU Days is then calculated by $(1.38/24) \times 4$, which is equal to 0.23. Moreover, in the "Search Method" column of Tables II and III, "MDM," "SMBO," and "MDL" stand for the "manually designed method," the "sequential model-based global optimization," and the "multinomial distribution learning," respectively. The results obtained from the proposed

TABLE II

COMPARISON OF MSNAS AND THE PEER COMPETITORS IN TERMS OF THE TEST ERROR RATES (MEAN ± STD | BEST) AND THE NUMBER OF WEIGHTS AND THE CONSUMED COMPUTATIONAL COST ON THE CIFAR10 DATASETS

Architecture	Test Error On C10	GPU Days	params	Search method
VGG [25]	6.66	-	28.05M	MDM
ResNet-110 [3]	6.61±0.16 | 6.43	-	1.7M	MDM
WRN [27]	4.00	-	36.25M	MDM
SENet [26]	4.62	-	10.2M	MDM
ResNetXt-29 [28]	3.58	-	68.1M	MDM
DenseNet-BC [5]	5.19	-	15.3M	MDM
MobileNetV2 [29]	5.44	-	2.1M	MDM
ShuffleNet [85]	9.13	-	1.06M	MDM
NASNet-A [24]	3.41	22400	3.3M	RL
MetaQNN [32]	6.92	100	-	RL
ENAS [30]	2.89	0.45	4.6M	RL
ProxylessNAS [31]	2.08	1500	5.7M	RL
BNAS-CCE [33]	2.88	0.19	4.8M	RL
PNASNet [54]	3.41±0.09 | 3.32	150	3.2M	SMBO
NAS-BOWL [47]	2.50	3	3.7M	BO
MdeNAS [74]	2.55	0.16	3.61M	MDL
TE-NAS [48]	2.63	0.05	3.8M	Training-Free
DARTS [16]	2.72	1	3.4M	GD
P-DARTS [34]	2.50	0.3	3.4M	GD
PC-DARTS [17]	2.57	0.1	3.6M	GD
SNAS [35]	2.85±0.02 | 2.83	1.5	2.8M	GD
Firefly [36]	2.78±0.05 | 2.73	1.5	3.3M	GD
AdaptNAS-S [37]	2.50	2	5.3M	GD
GibbsNAS+cutout [38]	2.53	0.5	4.1M	GD
BPC-DARTS [39]	2.57	0.1	3.6M	GD
SGAS+PT [40]	2.46	0.29	3.9M	GD
Large-Scale Evo [19]	4.4	2750	40.4	EA
Hierarchical Evo [42]	3.63	300	15.7M	EA
AmoebaNet-B [21]	2.55	3150	2.8M	EA
LEMONADEADE [43]	2.58	90	13.1M	EA
NSGANet [44]	2.75	4	3.3M	EA
AE-CNN [45]	4.3	27	2.0M	EA
CARS [23]	2.62	0.4	3.6M	EA
SI-EvoNet-S [41]	2.69	0.458	1.84M	EA
MSNAS(ours)	**2.68±0.08 | 2.53**	**0.23**	**3.25M**	**EA**

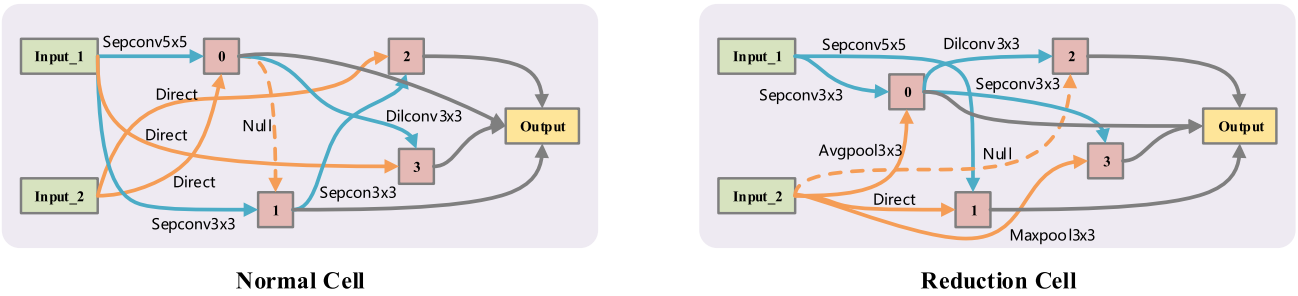


Fig. 9. Illustration of the best normal cell and reduction cell found by our proposed MSNAS on the CIFAR10 dataset. In each cell, the blue and yellow solid lines represent the convolution operators and pooling operators, respectively. The yellow dashed lines describe that there is no connection between the two corresponding neural nodes. The gray solid lines indicate the concatenation of the information flows at the target node.

algorithm are highlighted in bold, which are presented at the last rows of Tables II and III.

As can be observed in Table II, the proposed method obtained superior performance in terms of both best test errors against all the manually designed methods for CNN architecture design on CIFAR10. In particular, the proposed MSNAS achieved the best classification error of approximately 1.0%–6.5% lower than the manually designed CNN architectures on the CIFAR10 dataset. In contrast to the nonevolutionary NAS algorithms, the proposed MSNAS obtained superior and competitive best and average test error performance. Specifically, among totally 18 nonevolutionary NAS methods, MSNAS achieved better best and average test error over 11 methods, i.e., NASNet-A, MetaQNN, ENAS, PNASNet, MdeNAS, DARTS, PC-DARTS, BNAS-CCE, SNAS, BPC-DARTS, TE-NAS, and SNAS. Moreover, it is observed that, although the proposed MSNAS obtained a slightly worse best test error compared to ProxylessNAS (−0.45%), P-DARTS (−0.03%), AdaptNAS-S (−0.03%), SGAS +PT (−0.07%), and NAS-BOWL (−0.03%), MSNAS used the least time,

TABLE III

COMPARISON OF MSNAS AND THE PEER COMPETITORS IN TERMS OF THE BEST TEST ERROR RATES, THE AVERAGE TEST ERROR RATES, THE NUMBER OF WEIGHTS, AND THE CONSUMED COMPUTATIONAL COST CIFAR100 DATASETS

Architecture	Best Test Error	GPU Days	params	Search method
VGG [25]	32.05	-	28.05M	MDM
ResNet-110 [3]	25.16	-	1.7M	MDM
WRN [27]	19.25	-	36.25M	MDM
SENet [26]	22.71	-	10.2M	MDM
ResNetXt-29 [28]	17.31	-	68.1M	MDM
DenseNet-BC [5]	19.64	-	15.3M	MDM
MobileNetV2 [29]	22.91	-	2.1M	MDM
ShuffleNet [85]	22.86	-	1.06M	MDM
MetaQNN [32]	17.14	100	-	RL
PNASNet [54]	17.20	150	3.2M	SMBO
P-DARTS [34]	17.20	0.3	3.4M	GD
Large-Scale Evo [19]	23	2750	40.4M	EA
AE-CNN [45]	20.85	36	5.4M	EA
SI-EvoNet-S [41]	15.7	0.813	3.32M	EA
MSNAS (transfer)	**17.20**	**0.23**	**3.25M**	**EA**
MSNAS (search)	**16.41**	**0.25**	**3.19M**	**EA**

and the number of parameters of the final architecture is the smallest in contrast to these methods. In addition, MSNAS used less GPU days and parameters over most of the methods and slightly worse than BNAS-CCE (-0.04), MdeNAS (-0.07), TE-NAS (-0.18), PC-DARTS (-0.13), and BPC-DARTS (-0.13). Finally, for the third category of the compared method, by using the least of GPU days, the proposed MSNAS achieved the lowest best test error on CIFAR10. The best CNN architecture found by our proposed method on CIFAR10 is illustrated in Fig. 9.

Next, on CIFAR 100, there are two types of results are obtained by the proposed method: one is obtained by transferring the architecture found on CIFAR 10 [labeled as MSNAS (transfer)], and the other is obtained by performing the proposed MANAS on CIFAR 100 directly [labeled as MSNAS (transfer)]. For the compared algorithms, except the manually designed methods, there are also two types of results obtained by the algorithms on CIFAR 100. In particular, the results obtained by the nonevolutionary NAS methods and large-scale Evo are using transferred architectures found on the CIFAR 10 dataset by these methods, which are referred to their published papers. On the other hand, the results of AE-CNN and SI-EvoNet-S are obtained by the CNN architectures found on CIFAR 100 directly, which are also referred to the corresponding published papers. As can be observed from Table III, in contrast to the manually designed methods, our proposed MSNAS obtained the lowest best test error even using the architecture found on CIFAR 10 [i.e., see the results obtained by MSNAS (transfer)]. Furthermore, MSNAS obtained competitive best test error over the nonevolutionary NAS methods. However, MSNAS used fewer GPU days over all the three nonevolutionary NAS methods. Finally, using both fewer GPU days and less number of parameters, the proposed MSNAS also obtained superior or competitive best test error when compared to the EA-based NAS method for automated CNN architecture design. Particularly, MSNAS(search) obtained the classification error that is 4.44% lower than AE-CNN and over 6.5% lower than large-scale Evo. Although the best test error of MSNAS(search) is slightly worse than SI-EvoNet-S (-0.69%), MSNAS(search) is approximately three times faster than SI-EvoNet-S. The number of weights in MSNAS(search) is 3.19 M, while the number of weights in SI-EVoNet-S is 3.25 M.

TABLE IV

COMPARISON ABOUT THE EFFECT OF DIFFERENT SIMPLIFIED STRUCTURES AND SEARCH SPACES TO THE EVALUATION STRATEGY IN TERMS OF THE TEST ERROR RATES ON THE CIFAR10 DATASETS

Scheme	Average Test Error
NR & CP	3.025
NRN & CP	3.005
NRNRN & CP	**2.679**
CPU & NRNRN	2.927
CP & NRNRN	**2.679**

In addition, we further conducted investigations on the simplified structure proposed in the one-epoch-based performance evaluation, as discussed in Section III-C. In particular, let "N" and "R" represent the normal cell and the reduction cell, respectively. Besides the proposed simplified structure "NRNRN," the other two common simplified structures "NR" and "NRN" are considered. Furthermore, as the inputs of each neural node may affect the performance of the simplified structure, we also conduct investigations on the effectiveness of the proposed search method that considers convolution and pooling operators separately for each neural node. Table IV summarizes the investigation results of the proposed MSNAS with different simplified structures in the one-epoch-based performance evaluation and different search schemes for selecting input of each neural node with the same simplified structures on the CRIFAR 10 datasets over five independent runs. In Table IV, "CP" denotes the proposed method in which the two inputs of a neural node are from the convolution operators and pooling operators, respectively. On the other hand, "CPU" denotes the common cell-based space where each input of a neural node can be processed by either convolution operators or pooling operators. Therefore, the first column of Table IV gives all the combinations of search schemes and the simplified structures. It can be observed from Table IV that, with the same search scheme, the proposed simplified structure "NRNRN" obtained the lowest average test error. Using the same simplified structure for performance evaluation, the proposed search scheme in MSNAS obtained supervisor average test error against the common search scheme considered for cell-based search space. These again confirmed the effectiveness of the proposed MSNAS for automated CNN architecture design.

Specifically, from the results shown in Table IV, we can see there are two observations. First, the stable number of model weights can facilitate the comparison of model performance (the experiments of the two schemes "CPU & NRNRN" and "CP & NRNRN"), which means that, with the same setting of training times, the numbers of weight updates for different models are similar. It, thus, may be more effective to compare two models in the initial training epoch. Second, during the search process, the simpler the simplified model that is a

stacking of normal cell and reduction cell for comparing model performance, the poorer the search performance (the experiments about three schemes "NR & CP," "NRN & CP," and "NRNRN & CP"). This demonstrates that oversimplification of the model is unfriendly for the comparison of model performance, and it will lead to degradation of search performance. Therefore, the rationale behind the one-epoch evaluation strategy could be the stable number of model weights and the design of a suitable simplified model.

V. Conclusion and Future Work

In this article, we have proposed an efficient memetic search-based NAS algorithm for automated CNN architecture design. In particular, the details of the proposed search space that divides the original cell-based search space into convolution space and pooling space, the search strategy, including global search and local search, and the one-epoch-based estimation strategy for evaluating a particular CNN architecture are presented and discussed. To evaluate the performance of the proposed method, comprehensive studies against 34 state-of-the-art NAS methods for CNN, including manually designed methods, nonevolutionary methods, and evolutionary methods, on the commonly used CIFAR 10 and CIFAR 100 datasets are conducted. The obtained results in terms of best test error, average test error, GPU days, and the number of parameters confirmed the efficiency and effectiveness of the proposed method for automated CNN architecture design in cell-based space.

For future work, we would like to further investigate the design of search space and the one-epoch-based performance estimation strategy. For the design of search space, we would like to explore whether this idea of space splitting can be extended to general search spaces. Second, the proposed one-epoch-based performance estimation strategy can significantly reduce the computational overhead. In the future, we would also like to further conduct a comprehensive investigation on the influence of different space restrictions in the performance evaluation strategy and extend the evaluation strategy to general search spaces.

References

[1] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," in *Proc. Adv. Neural Inf. Process. Syst. (NIPS)*, vol. 25. Stateline, NV, USA, Dec. 2012, pp. 1097–1105.

[2] C. Szegedy *et al.*, "Going deeper with convolutions," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2015, pp. 1–9.

[3] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 770–778.

[4] G. Larsson, M. Maire, and G. Shakhnarovich, "FractalNet: Ultra-deep neural networks without residuals," 2016, *arXiv:1605.07648*.

[5] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 4700–4708.

[6] G. Ghiasi, T.-Y. Lin, and Q. V. Le, "NAS-FPN: Learning scalable feature pyramid architecture for object detection," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 7036–7045.

[7] Y. Chen, T. Yang, X. Zhang, G. Meng, X. Xiao, and J. Sun, "DetNAS: Backbone search for object detection," 2019, *arXiv:1903.10979*.

[8] C. Liu *et al.*, "Auto-DeepLab: Hierarchical neural architecture search for semantic image segmentation," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 82–92.

[9] V. Nekrasov, H. Chen, C. Shen, and I. Reid, "Fast neural architecture search of compact semantic segmentation models via auxiliary cells," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 9126–9135.

[10] V. Nekrasov, H. Chen, C. Shen, and I. Reid, "Architecture search of dynamic cells for semantic video segmentation," in *Proc. IEEE Winter Conf. Appl. Comput. Vis. (WACV)*, Mar. 2020, pp. 1970–1979.

[11] N. Klyuchnikov, I. Trofimov, E. Artemova, M. Salnikov, M. Fedorov, and E. Burnaev, "NAS-bench-NLP: Neural architecture search benchmark for natural language processing," 2020, *arXiv:2006.07116*.

[12] A. MacLaughlin, J. Dhamala, A. Kumar, S. Venkatapathy, R. Venkatesan, and R. Gupta, "Evaluating the effectiveness of efficient neural architecture search for sentence-pair tasks," in *Proc. 1st Workshop Insights From Negative Results (NLP)*, 2020, pp. 1–10.

[13] P. Ren *et al.*, "A comprehensive survey of neural architecture search: Challenges and solutions," 2021, *arXiv:2006.02903*.

[14] I. Bello, B. Zoph, V. Vasudevan, and Q. V. Le, "Neural optimizer search with reinforcement learning," in *Proc. 34th Int. Conf. Mach. Learn.*, vol. 70, 2017, pp. 459–468.

[15] H. Liu, K. Simonyan, and Y. Yang, "Darts: Differentiable architecture search," 2019, *arXiv:1806.09055*.

[16] Y. Xu *et al.*, "PC-DARTS: Partial channel connections for memory-efficient architecture search," in *Proc. Int. Conf. Learn. Represent.*, 2020, pp. 1–12. [Online]. Available: https://openreview.net/forum?id=BJlS634tPr

[17] K. O. Stanley and R. Miikkulainen, "Evolving neural networks through augmenting topologies," *Evol. Comput.*, vol. 10, no. 2, pp. 99–127, 2002.

[18] E. Real *et al.*, "Large-scale evolution of image classifiers," in *Proc. 34th Int. Conf. Mach. Learn.*, vol. 70, 2017, p. 2902–C2911.

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

[20] E. Real, A. Aggarwal, Y. Huang, and Q. V. Le, "Regularized evolution for image classifier architecture search," in *Proc. AAAI Conf. Artif. Intell.*, vol. 33, 2019, pp. 4780–4789.

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

[22] Z. Yang *et al.*, "CARS: Continuous evolution for efficient neural architecture search," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2020, pp. 1829–1838.

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

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

[25] K. He, X. Zhang, S. Ren, and J. Sun, "Identity mappings in deep residual networks," in *Proc. Eur. Conf. Comput. Vis.*, Cham, Switzerland: Springer, 2016, pp. 630–645.

[26] S. Zagoruyko and N. Komodakis, "Wide residual networks," 2016, *arXiv:1605.07146*.

[27] S. Xie, R. Girshick, P. Dollar, Z. Tu, and K. He, "Aggregated residual transformations for deep neural networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 1492–1500.

[28] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "MobileNetV2: Inverted residuals and linear bottlenecks," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 4510–4520.

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

[30] H. Cai, L. Zhu, and S. Han, "ProxylessNAS: Direct neural architecture search on target task and hardware," 2018, *arXiv:1812.00332*.

[31] B. Baker, O. Gupta, N. Naik, and R. Raskar, "Designing neural network architectures using reinforcement learning," 2016, *arXiv:1611.02167*.

[32] Z. Ding, Y. Chen, N. Li, D. Zhao, Z. Sun, and C. L. P. Chen, "BNAS: Efficient neural architecture search using broad scalable architecture," *IEEE Trans. Neural Netw. Learn. Syst.*, early access, Mar. 31, 2021, doi: 10.1109/TNNLS.2021.3067028.

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

[34] S. Xie, H. Zheng, C. Liu, and L. Lin, "SNAS: Stochastic neural architecture search," 2018, *arXiv:1812.09926*.

[35] L. Wu, B. Liu, P. Stone, and Q. Liu, "Firefly neural architecture descent: A general approach for growing neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 33, 2020, pp. 22373–22383.

[36] Y. Li, Z. Yang, Y. Wang, and C. Xu, "Adapting neural architectures between domains," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 33, 2020, pp. 789–798.

[37] C. Xue, X. Wang, J. Yan, Y. Hu, X. Yang, and K. Sun, "Rethinking bi-level optimization in neural architecture search: A Gibbs sampling perspective," in *Proc. AAAI Conf. Artif. Intell.*, vol. 35, no. 12, 2021, pp. 10551–10559.

[38] Y. Xu *et al.*, "Partially-connected neural architecture search for reduced computational redundancy," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 43, no. 9, pp. 2953–2970, Sep. 2021.

[39] R. Wang, M. Cheng, X. Chen, X. Tang, and C.-J. Hsieh, "Rethinking architecture selection in differentiable NAS," 2021, *arXiv:2108.04392*.

[40] H. Zhang, Y. Jin, R. Cheng, and K. Hao, "Efficient evolutionary search of attention convolutional networks via sampled training and node inheritance," *IEEE Trans. Evol. Comput.*, vol. 25, no. 2, pp. 371–385, Apr. 2021.

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

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

[43] Z. Lu *et al.*, "NSGA-Net: Neural architecture search using multi-objective genetic algorithm," in *Proc. Genetic Evol. Comput. Conf.*, Jul. 2019, pp. 419–427.

[44] Y. Sun, B. Xue, M. Zhang, and G. G. Yen, "Completely automated CNN architecture design based on blocks," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 31, no. 4, pp. 1242–1254, Apr. 2020.

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

[46] B. Ru, X. Wan, X. Dong, and M. Osborne, "Interpretable neural architecture search via Bayesian optimisation with Weisfeiler–Lehman kernels," 2020, *arXiv:2006.07556*.

[47] W. Chen, X. Gong, and Z. Wang, "Neural architecture search on ImageNet in four GPU hours: A theoretically inspired perspective," 2021, *arXiv:2102.11535*.

[48] T. Elsken, J. H. Metzen, and F. Hutter, "Neural architecture search: A survey," 2018, *arXiv:1808.05377*.

[49] B. Zoph and Q. V. Le, "Neural architecture search with reinforcement learning," 2016, *arXiv:1611.01578*.

[50] M. Tan *et al.*, "MnasNet: Platform-aware neural architecture search for mobile," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 2820–2828.

[51] H. Hu, J. Langford, R. Caruana, E. Horvitz, and D. Dey, "Macro neural architecture search revisited," in *Proc. 2nd Workshop Meta-Learn. (NeurIPS)*, 2018, pp. 1–6.

[52] 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 *Proc. Eur. Conf. Comput. Vis. (ECCV)*, Sep. 2018, pp. 517–531.

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

[54] Z. Zhong, J. Yan, W. Wu, J. Shao, and C.-L. Liu, "Practical block-wise neural network architecture generation," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 2423–2432.

[55] Z. Zhong *et al.*, "BlockQNN: Efficient block-wise neural network architecture generation," 2018, *arXiv:1808.05584*.

[56] L. Li and A. Talwalkar, "Random search and reproducibility for neural architecture search," in *Proc. Uncertainty Artif. Intell.*, 2020, pp. 367–377.

[57] A. Brock, T. Lim, J. M. Ritchie, and N. Weston, "SMASH: One-shot model architecture search through HyperNetworks," 2017, *arXiv:1708.05344*.

[58] H. Liang *et al.*, "DARTS+: Improved differentiable architecture search with early stopping," 2019, *arXiv:1909.06035*.

[59] L. Li, K. G. Jamieson, G. DeSalvo, A. Rostamizadeh, and A. Talwalkar, "Hyperband: Bandit-based configuration evaluation for hyperparameter optimization," in *Proc. ICLR (Poster)*, 2017.

[60] E. Real, A. Aggarwal, Y. Huang, and Q. V. Le, "Aging evolution for image classifier architecture search," in *Proc. AAAI Conf. Artif. Intell.*, vol. 3, 2019.

[61] H. Cai, T. Chen, W. Zhang, Y. Yu, and J. Wang, "Efficient architecture search by network transformation," 2017, *arXiv:1707.04873*.

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

[63] P. Moscato and M. G. Norman, "A memetic approach for the traveling salesman problem implementation of a computational ecology for combinatorial optimization on message-passing systems," *Parallel Comput. Transputer Appl.*, vol. 1, pp. 177–186, Sep. 1992.

[64] N. J. Radcliffe and P. D. Surry, "Formal memetic algorithms," in *Proc. AISB workshop Evol. Comput.*, Cham, Switzerland: Springer, 1994, pp. 1–16.

[65] Y. S. Ong, N. Krasnogor, and H. Ishibuchi, "Special issue on memetic algorithms," *IEEE Trans. Syst., Man, B, Cybern.*, vol. 37, no. 1, pp. 2–5, Feb. 2007.

[66] L. Feng, Y.-S. Ong, M.-H. Lim, and I. W. Tsang, "Memetic search with interdomain learning: A realization between CVRP and CARP," *IEEE Trans. Evol. Comput.*, vol. 19, no. 5, pp. 644–658, Oct. 2015.

[67] A. Gupta and Y.-S. Ong, *Memetic Computation: The Mainspring of Knowledge Transfer in a Data-Driven Optimization Era*, vol. 21. Cham, Switzerland: Springer, 2018.

[68] G. Li, Z. Zhu, L. Ma, and X. Ma, "Multi-objective memetic algorithm for core-periphery structure detection in complex network," *Memetic Comput.*, vol. 13, no. 3, pp. 285–306, Sep. 2021.

[69] B. Liu, L. Wang, and Y. H. Jin, "An effective PSO-based memetic algorithm for flow shop scheduling," *IEEE Trans. Syst., Man, B, Cybern.*, vol. 37, no. 1, pp. 18–27, Feb. 2007.

[70] J. Tang, M. H. Lim, and Y. S. Ong, "Parallel memetic algorithm with delective local search for large scale quadratic assignment problems," *Int. J. Innov. Comput., Inf. Control*, vol. 2, no. 6, pp. 1399–1416, 2006.

[71] K. Tang, Y. Mei, and X. Yao, "Memetic algorithm with extended neighborhood search for capacitated arc routing problems," *IEEE Trans. Evol. Comput.*, vol. 13, no. 5, pp. 1159–1166, 2009.

[72] Z. Zhu, Y. S. Ong, and M. Dash, "Parallel memetic algorithm with delective local search for large scale quadratic assignment problems," *Wrapper-Filter Feature Selection Algorithm Using Memetic Framework*, vol. 37, no. 1, pp. 70–76, 2007.

[73] X. Zheng, R. Ji, L. Tang, B. Zhang, J. Liu, and Q. Tian, "Multinomial distribution learning for effective neural architecture search," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2019, pp. 1304–1313.

[74] K. V. Price, "Differential evolution," in *Handbook of Optimization*. Cham, Switzerland: Springer, 2013, pp. 187–214.

[75] Y. Shang and B. W. Wah, "A discrete Lagrangian-based global-search method for solving satisfiability problems," *J. Global Optim.*, vol. 12, no. 1, pp. 61–99, 1998.

[76] D. Whitley, "A genetic algorithm tutorial," *Statist. Comput.*, vol. 4, no. 2, pp. 65–85, Jun. 1994.

[77] U. Maulik and S. Bandyopadhyay, "Genetic algorithm-based clustering technique," *Pattern Recognit.*, vol. 33, no. 9, pp. 1455–1465, 2000.

[78] G. R. Harik, F. G. Lobo, and D. E. Goldberg, "The compact genetic algorithm," *IEEE Trans. Evol. Comput.*, vol. 3, no. 4, pp. 287–297, Nov. 1999.

[79] S. Das and P. N. Suganthan, "Differential evolution: A survey of the state-of-the-art," *IEEE Trans. Evol. Comput.*, vol. 15, no. 1, pp. 4–31, Feb. 2011.

[80] A. Hameed and A. N. Mian, "Towards better traffic localization of virtual LANs using genetic algorithm," *Comput. J.*, vol. 59, no. 2, pp. 178–191, Feb. 2016.

[81] C. W. Ahn and R. S. Ramakrishna, "Elitism-based compact genetic algorithms," *IEEE Trans. Evol. Comput.*, vol. 7, no. 4, pp. 367–385, Aug. 2003.

[82] Y. Fang and J. Li, "A review of tournament selection in genetic programming," in *Advances in Computation and Intelligence*, Z. Cai, C. Hu, Z. Kang, and Y. Liu, Eds. Berlin, Germany: Springer, 2010, pp. 181–192.

[83] A. Krizhevsky *et al.*, "Learning multiple layers of features from tiny images," Tech. Rep., 2009.

[84] X. Zhang, X. Zhou, M. Lin, and J. Sun, "ShuffleNet: An extremely efficient convolutional neural network for mobile devices," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 6848–6856.

Junwei Dong received the B.E. degree from the College of Computer Science, Chongqing University, Chongqing, China, in 2018, where he is currently pursuing the Ph.D. degree.

His current research interests include neural architecture search, deep learning, and evolutionary transfer optimization.

Boyu Hou received the B.E. degree from the College of Computer Science, Chongqing University, Chongqing, China, in 2019, where he is currently pursuing the Ph.D. degree.

His current research interests include neural architecture search, knowledge distillation, and evolutionary transfer optimization.

Liang Feng received the Ph.D. degree from the School of Computer Engineering, Nanyang Technological University, Singapore, in 2014.

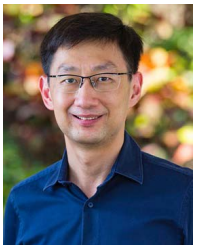
He is currently a Professor at the College of Computer Science, Chongqing University, Chongqing, China. His research interests include computational and artificial intelligence, memetic computing, big data optimization and learning, and transfer learning.

Dr. Feng's research work on evolutionary multitasking received the 2019 IEEE TRANSACTIONS ON EVOLUTIONARY COMPUTATION Outstanding Paper Award. He is the founding Chair of the IEEE CIS Intelligent Systems Applications Technical Committee Task Force on Transfer Learning and Transfer Optimization. He is also an Associate Editor of the *IEEE Computational Intelligence Magazine*, *Memetic Computing*, and *Cognitive Computation*.

Huajin Tang received the B.Eng. degree from Zhejiang University, Hangzhou, China, in 1998, the M.Eng. degree from Shanghai Jiao Tong University, Shanghai, China, in 2001, and the Ph.D. degree from the National University of Singapore, Singapore, in 2005.

He was a Research and Development Engineer with STMicroelectronics, Singapore, from 2004 to 2006. From 2006 to 2008, he was a Post-Doctoral Fellow with the Queensland Brain Institute, The University of Queensland, Brisbane, QLD, Australia. He was the Head of the Robotic Cognition Laboratory, Institute for Infocomm Research, Singapore, from 2008 to 2015. Since 2014, he has been a Professor and the Director of the Neuromorphic Computing Research Center, Sichuan University, Chengdu, China. He is currently a Professor with Zhejiang University. His research interests include neuromorphic computing, neuromorphic hardware and cognitive systems, and robotic cognition.

Dr. Tang is a Board of Governor Member of the International Neural Network Society. His research work on Brain GPS has been reported by *MIT Technology Review* and *Communications of the ACM*. He received the 2011 Role Model Award of Institute for Infocomm Research Singapore, the 2016 IEEE TRANSACTIONS ON NEURAL NETWORKS AND LEARNING SYSTEMS Outstanding Paper Award, and the 2019 *IEEE Computational Intelligence Magazine* Outstanding Paper Award. He has served as an Associate Editor of the IEEE TRANSACTIONS ON NEURAL NETWORKS AND LEARNING SYSTEMS, *Frontiers in Neuromorphic Engineering*, and *Neural Networks*. He is currently the Editor-in-Chief of IEEE TRANSACTIONS ON COGNITIVE AND DEVELOPMENTAL SYSTEMS.



Kay Chen Tan (Fellow, IEEE) received the B.Eng. (Hons.) and Ph.D. degrees from the University of Glasgow, Glasgow, U.K., in 1994 and 1997, respectively.

He is currently a Chair Professor of computational intelligence at the Department of Computing, The Hong Kong Polytechnic University, Hong Kong. He has published over 300 refereed articles and seven books.

Dr. Tan is currently the Vice-President (Publications) of the IEEE Computational Intelligence Society, USA. He served as the Editor-in-Chief of the *IEEE Computational Intelligence Magazine* from 2010 to 2013 and the IEEE TRANSACTIONS ON EVOLUTIONARY COMPUTATION from 2015 to 2020. He currently serves as an editorial board member for more than ten journals. He is an IEEE Distinguished Lecturer Program (DLP) Speaker and the Chief Coeditor of Springer Book Series on *Machine Learning: Foundations, Methodologies, and Applications*.

Yew-Soon Ong (Fellow, IEEE) received the Ph.D. degree in artificial intelligence in complex design from the University of Southampton, Southampton, U.K., in 2003.

He is currently the President's Chair Professor in computer science at Nanyang Technological University (NTU), Singapore, and holds the position of Chief Artificial Intelligence Scientist at the Agency for Science, Technology and Research, Singapore. At NTU, he serves as the Co-Director of the Singtel-NTU Cognitive and Artificial Intelligence Joint Laboratory. His research interest is in artificial and computational intelligence.

Dr. Ong is the founding Editor-in-Chief of the IEEE TRANSACTIONS ON EMERGING TOPICS IN COMPUTATIONAL INTELLIGENCE and an Associate Editor of IEEE TRANSACTIONS ON NEURAL NETWORKS AND LEARNING SYSTEMS, IEEE TRANSACTIONS ON CYBERNETICS, IEEE TRANSACTIONS ON ARTIFICIAL INTELLIGENCE, and others. He has received several IEEE outstanding paper awards and was listed as a Thomson Reuters Highly Cited Researcher and among the World's Most Influential Scientific Minds.