Complex & Intelligent Systems (2023) 9:213–231
https://doi.org/10.1007/s40747-022-00794-7

ORIGINAL ARTICLE

Architecture entropy sampling-based evolutionary neural architecture search and its application in osteoporosis diagnosis

Jianjun Chu[1] · Xiaoshan Yu[2,3] · Shangshang Yang[2,3] · Jianfeng Qiu[2,3] · Qijun Wang[2,3]

Received: 21 January 2022 / Accepted: 27 May 2022 / Published online: 25 June 2022

Abstract

In recent years, neural architecture search (NAS) has achieved unprecedented development because of its ability to automatically achieve high-performance neural networks in various tasks. Among these, the evolutionary neural architecture search (ENAS) has impressed the researchers due to the excellent heuristic exploration capability. However, the evolutionary algorithm-based NAS are prone to the loss of population diversity in the search process, causing that the structure of the surviving individuals is exceedingly similar, which will lead to premature convergence and fail to explore the search space comprehensively and effectively. To address this issue, we propose a novel indicator, named architecture entropy, which is used to measure the architecture diversity of population. Based on this indicator, an effective sampling strategy is proposed to select the candidate individuals with the potential to maintain the population diversity for environmental selection. In addition, an unified encoding scheme of topological structure and computing operation is designed to efficiently express the search space, and the corresponding population update strategies are suggested to promote the convergence. The experimental results on several image classification benchmark datasets CIFAR-10 and CIFAR-100 demonstrate the superiority of our proposed method over the state-of-the-art comparison ones. To further validate the effectiveness of our method in real applications, our proposed NAS method is applied in the identification of lumbar spine X-ray images for osteoporosis diagnosis, and can achieve a better performance than the commonly used methods. Our source codes are available at https://github.com/LabyrinthineLeo/AEMONAS.

Keywords Neural architecture search · Convolutional neural networks · Evolutionary algorithms · Architecture entropy · Medical image recognition

Introduction

Deep convolutional neural networks (CNNs) have made unprecedented progress in solving various computer vision tasks, such as image classification [24], semantic segmentation [18], object detection [28], and so on. Contributing to this success is the emergence of a large number of excellent CNN architectures, which include DenseNet [11], ResNet [8], GoogLe-Net [35], ShuffleNet [47], among many others. These efficient network models were manually designed by human experts with extensive expertise and experience, and these processes were labor-intensive and error-prone. Hence, it is very difficult for beginners to manually design their own network architecture according to their actual needs. To address this problem, automated neural architecture design, i.e., neural architecture search (NAS) has been proposed and rapidly developed. NAS aims to design neural architectures with excellent performance using constrained

✉ Qijun Wang
wangqijun308@163.com

Jianjun Chu
chujianjun@163.com

Xiaoshan Yu
e20201115@stu.ahu.edu.cn

Shangshang Yang
yangshang0308@gmail.com

Jianfeng Qiu
qiujianf@ahu.edu.cn

[1] The Second People's Hospital of Hefei, Hefei 230012, China

[2] Key Laboratory of Intelligent Computing and Signal Processing of Ministry of Education, Anhui University, Hefei 230039, China

[3] School of Artificial Intelligence, Anhui University, Hefei 230601, China

computing resources in an automatic manner without much human interaction. The work of NAS [52] and MetaQNN [2] is considered to be the pioneering work of neural architecture search, in which the reinforcement learning [34] method was adopted as an optimizer to search convolutional neural networks, and the architectures they obtained achieve remarkably competitive performance on the image classification tasks compared to the excellent manually designed models.

In general, NAS can be viewed as a complex optimization problem involving various aspects, such as the design of search space, resource constraints, and the objectives to be optimized. Existing NAS works can be divided into three categories according to the optimization algorithm employed, i.e., reinforcement learning (RL) based NAS [3,25], gradient-based NAS [17,21], and evolutionary algorithm (EA)-based NAS [19,32,33]. RL-based NAS algorithms mainly use recurrent neural network (RNN) [44] as a controller to sample the representation of neural architecture, and then adopt reinforcement learning as the corresponding search method to constantly adjust the network architectures. For the gradient-based NAS algorithms, they continuously relax the previously discrete search space, allowing efficient search of neural architectures using the gradient descent strategy, which significantly improves the search efficiency compared with RL-based NAS algorithms. Different from the former two approaches, the evolutionary algorithm-based NAS approach (ENAS) is a population-based heuristic search algorithm, where the network architecture is expressed as an individual by adopting a coding scheme, and the best neural architecture is obtained using evolutionary algorithms.

In recent years, ENAS has attracted increasing attention due to its powerful capability to achieve globally optimal solutions. Genetic CNN [40] proposed a new encoding mechanism, which encodes the topological information of neural network architectures into a fixed-length binary string. Afterward, hierarchical encoding [16] and variable-length encoding [32] were presented to improve the expressing ability. Considering that except for the classification accuracy, the complexity (parameter numbers) is another importance factor which should be taken into account, the multi-objective optimization methods were introduced into ENAS, and various optimal solutions can be achieved to adapt to different application situations. As for convolutional neural network, the minor difference between two individuals usually brings little change in fitness values. Thus, if population diversity is not maintained well during evolution, premature convergence will occur, and individuals collapse into copies of the same genotype before the search space is properly explored. Besides, due to the strong correlation between individual encoding and network architecture, inappropriate genetic operators (e.g., crossover, mutation, etc.) will make ENAS get trapped in local search space during the search process, and this is another reason for premature convergence caused by the lack of population diversity.

In this paper, we propose an architecture entropy-based multi-objective NAS, termed AEMONAS, to solve the aforementioned challenges, which can obtain more competitive neural networks. Meanwhile, to enlarge the search space and deeply explore the relationship between the topological structure of connections and the computing operations, the unified encoding scheme of topological structure and computing operation is suggested, and the corresponding genetic operators tailored for this encoding scheme are adopted to speedup the convergence. The main contributions of the proposed AEMONAS are summarized as follows:

- An indicator named architecture entropy is defined to measure the architecture diversity of the population. Specifically, the density value of each individual is calculated by parsing the encoding information, and the probability of each individual is derived through counting the number of individuals located at the same range of interval. Then, the entropy of the interval distribution of all individuals is calculated to measure the diversity of the population. Furthermore, a sampling strategy based on this indicator is proposed, and adopted at the stage of offspring generation in the evolutionary process to obtain individuals with the potential to maintain the diversity of offspring population.
- An unified encoding strategy of combining structural topology and computing operation of the neural architecture is proposed, and the novel encoding scheme can accommodate various basic computing units including the attention module, and the integration of connection structure and computing operation can enrich the search space to avoid the reduction of population diversity. Meanwhile, the genetic operators tailored for the novel encoding scheme are designed to improve the optimization efficiency of the evolutionary algorithm.
- The experimental results on several image classification benchmark datasets CIFAR-10 and CIFAR-100 demonstrate the superiority of our proposed method over the state-of-the-art comparison ones. In addition, to further validate the effectiveness of our method in real applications, our proposed NAS method is applied in the identification of lumbar spine X-ray images for osteoporosis diagnosis, and can achieve a better performance than the commonly used methods.

The rest of this article is organized as follows. We first describe the related works and then describe the searching space of our proposed method. Next, we depict the details of the proposed AEMONAS. Subsequently, we present and analyze the experimental results, and then report the appli-

cation of AEMONAS in osteoporosis diagnosis. Finally, we conclude this paper and looks into the future work.

Related work

In this section, we will first introduce the problem definition of evolutionary neural architecture search (ENAS), and then briefly introduce the classical ENAS algorithms which are closely related to our work. Finally, we summarize the existing works of ENAS, especially the works on maintaining the population diversity.

In general, the task of neural architecture search for a given dataset $D = \{D_{train}, D_{valid}, D_{test}\}$ and the corresponding search space A can be formulated as [17]

$$\begin{aligned} &\min\ F(\alpha, \omega^*) \\ &\text{s.t. } \omega^* \in \arg\min_{\omega}\ L_{train}(\alpha, \omega), \end{aligned} \tag{1}$$

where $\alpha \in A$ represents the network architecture to be optimized, and ω and ω^* denote the corresponding weights of α and the optimal weights after training on the corresponding dataset D_{train}, respectively, and L_{train} denotes the loss function on D_{train}. Early NAS works usually define a single-objective optimization problem for searching the network architecture with emphasis on accuracy, i.e., $F(\alpha, \omega^*) = Error_{valid}(\alpha, \omega^*)$, where $Error_{valid}$ denotes the error rate on the validation dataset. However, with the rise of various terminal devices with resource constraints, the complexity of the architecture has become an important issue to be considered, and then, the NAS problem becomes a multi-objective optimization problem [6,19,20]

$$F(\alpha, \omega^*) = \begin{cases} Error_{valid}(\alpha, \omega^*) \\ Complexity(\alpha), \end{cases} \tag{2}$$

where $Complexity(\alpha)$ represents the complexity of architecture α. In this paper, we use the size of model parameters to quantify this metric of complexity. For this problem, many methods can be applied to solve it, such as analytical methods, gradient-based methods, etc. The evolutionary algorithms have the powerful capability to approximate the globally optimal solutions, and can be utilized to conduct the NAS task.

Evolutionary neural architecture search algorithms (ENAS) refer to adopting evolutionary algorithms [4] as the search strategy to optimize the NAS problem. Due to the excellent ability to tackle complex combinatorial optimization problems of evolutionary algorithms, many creative ENAS algorithms have emerged in recent years. Large-scale evolution [26] employed an evolutionary strategy at enormous scales to explore network architectures for CIFAR-10 and CIFAR-100, resulting in classification error rates of 5.4% and 23.0%, respectively. Due to the unprecedented scale of evolution, the search process of this algorithm took about 2750 GPU days. Genetic-CNN [40] proposed a new encoding mechanism, which encoded the topological information of neural network architectures into a fixed-length binary string, greatly improving the expression ability of search space, and then adopted the genetic algorithm to evolve offspring population to generate competitive architecture individuals. Hierarchical evolution algorithm [16] designed a novel hierarchical genetic expression mechanism through the combination of modularized design, which greatly enriched the search space and supported more complex topologies. EvoCNN [32] proposed variable-length encoding strategy and used an improved EA to automatically evolve the architecture of neural networks. Regularized evolution [27] introduced an evolutionary strategy based on age properties to enhance environmental selection, and finally reached 83.9% top-1 accuracy for ImageNet. NSGA-Net [19] used a multi-objective evolutionary algorithm to evolve the population to discover various trade-off network architectures in terms of performance and complexity.

Population diversity [30] is one of the most important factors for achieving global and efficient search in evolutionary algorithms. In the long history of evolutionary computation, many methods have been proposed to maintain or promote population diversity [22,23,38]. In recent years, some researches have preliminarily explored some mechanisms to maintain the population diversity in ENAS. Hoa et al. [9] adopted well-tested moderate mutation rate to ensure population diversity. In [51], a hierarchical training strategy was proposed, and in this work, the mutated network architectures were allocated to different population sets, and sampling was taken from the population sets with different probabilities to realize the diversity of populations. Awad et al. [1] presented a modified canonical differential evolution (DE) for NAS and the DE is enabled to work on individuals from a uniform and continuous space, to prevent a dramatic drop of population diversity. Wei et al. [39] adopted an environmental selection mechanism that considered both elitism and diversity of population to improve search efficiency and simultaneously prevent premature convergence.

To sum up, ENAS has the potential to reach the best neural architecture. Due to the problem in maintaining population diversity, current ENAS algorithms usually suffer from the premature convergence. In the following, we will address this problem from the perspective of search space and offspring sampling to improve the performance of ENAS.

Architecture search space

The goal of neural architecture search algorithm is to obtain the optimal network architecture in the huge predefined search space, and so, the design and selection of architecture search space has a great impact on the results of NAS. In this section, we will describe the details about the adopted neural architecture search space and the proposed encoding scheme in our AEMONAS.

Search space

Referring to the previous works [15,41,53], we build the complete neural network architecture based on a hierarchical approach, where the basic cell structures are learned, and then, the basic cells are stacked together with a desired number of times to produce the final convolutional neural network, as shown in Fig. 1a. In this paper, the basic cells consist of normal cell and reduction cell. The stacking pattern of normal cell and reduction cell is fixed, and the purpose of NAS is to search the inner structure of normal cell and reduction cell.

Cell structure

The cell structure is essentially a neural block which could be represented by a directed acyclic graph consisting of nodes and edges. The node means the layer, and the edge indicates the connection. As illustrated in Fig. 1b, the circles represent nodes, and for each node, there is a specific computing operation, such as convolution, pooling, etc. The edges among nodes represent the layer connection in network. Each cell has two input nodes $h[-1]$ and $h[0]$, which represent the feature map output by the two precursor cells, respectively. Some nodes in cells are leaf nodes with no successor nodes, and their output will be concatenated as the output of the entire cell. Note that the difference between the normal cell and the reduction cell lies on the spatial resolution of the input and output feature map. The stride of all convolution operations in normal cell kept to 1, and the channel number of the input and output feature map is consistent. In contrast, the stride of reduction cell is set to 2, meaning that downsampling is conducted at the same time, and the number of the output feature map channels is doubled relative to that of the input.

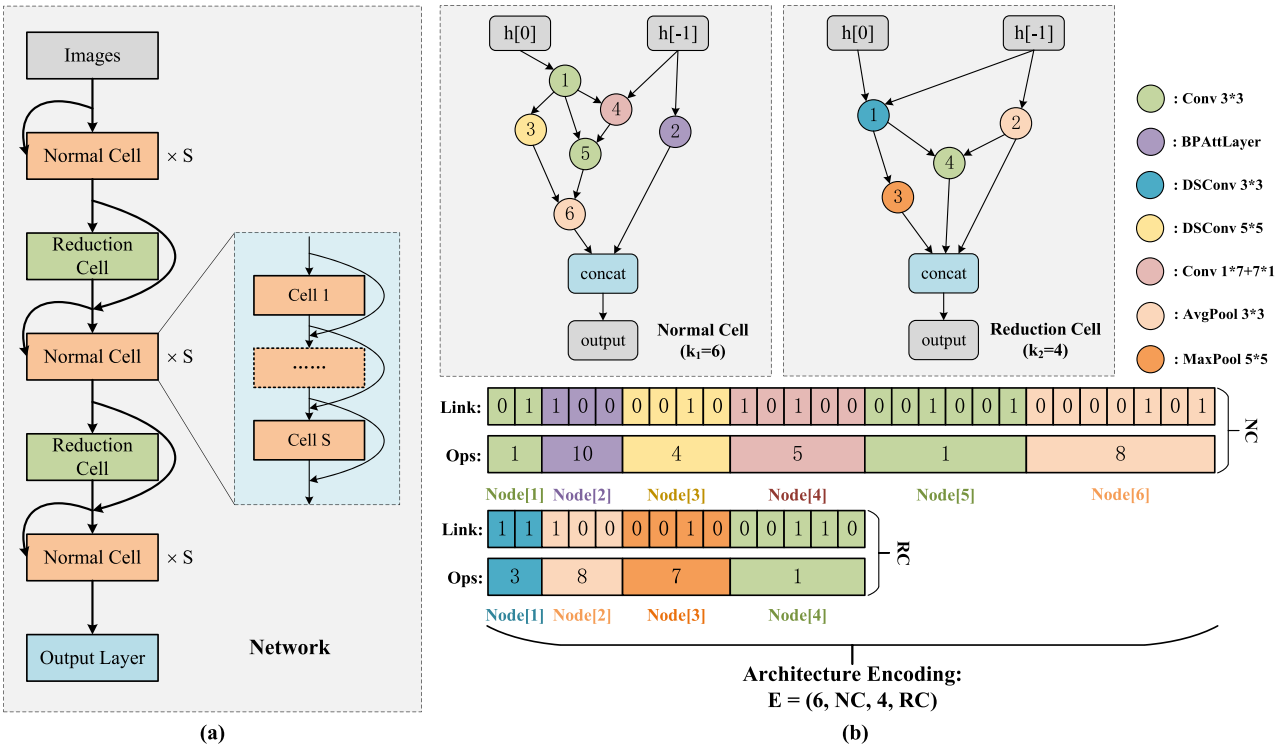


Fig. 1 The architecture search space and encoding scheme of the proposed AEMONAS. **a** The complete neural network architecture consisting of alternating stacks normal cell (NC) and reduction cell (RC), in which output layer is the classifier and Normal Cell repeat S times. **b** The two example basic cells represented by directed acyclic graphs (DAG) contain six and four nodes, respectively (i.e., normal cell and reduction cell), and the corresponding architecture encoding

Table 1 The operation space and corresponding interpretation

Operation	Explain	Index
Identity	Identity mapping	0
Conv 3*3	3*3 Vanilla Convolution	1
Conv 5*5	5*5 Vanilla Convolution	2
DSConv 3*3	3*3 Depthwise Separable Convolution	3
DSConv 5*5	5*5 Depthwise Separable Convolution	4
Conv 1*7+7*1	1*7 follow by 7*1 Convolution	5
MaxPool 3*3	3*3 Max Pooling	6
MaxPool 5*5	5*5 Max Pooling	7
AvgPool 3*3	3*3 Avg Pooling	8
AvgPool 5*5	5*5 Avg Pooling	9
BPAttLayer	Bilinear Pooling Attention Module	10

Once the basic cells are determined, we can stack them with a specified number of times to construct the entire network architecture. As shown in Fig. 1a, the entire network architecture is stacked by the normal cell and the reduction cell alternately. The interleaving pattern is the basic structure in the experiments, while the normal cell can be repeated S times. Note that the reduction cell is only responsible for downsampling to ensure that feature map can be spatially scaled down and the channel size is doubled, and so only one reduction cell is deployed at one time. Finally, the output layer of the entire neural network architecture is composed of the global average pooling layer and softmax layer, which can be adapted to different classification tasks. This approach is a common paradigm for successful hand-designed networks.

Node structure

The node is the basic unit which is used to construct the cell. The number of nodes contained in each cell is denoted by k and pre-set (e.g., k_1=6 (normal cell) and k_2=4 (reduction cell) in Fig. 1b). For current node, all the inputs are added together to form the ultimate input, and then, the computing operation bound to current node is applied to generate the output.

The computing operation of each node in the cells is all selected from the pre-configured operation collection of various classic and effective operations (e.g., depthwise separable convolution, average pooling, etc). As shown in Table 1, there are 11 operations in our operation collection. In the previous works [7,14], bilinear pooling operation was proposed to enhance the nonlinear modeling ability of convolutional neural network and learn higher dimensional information in feature maps. Inspired by them, we carefully design a bilinear pooling attention module named BPAttLayer to enrich the expression ability of the search space by combining bilinear pooling operation with channel attention mechanism. As shown in Fig. 2, the proposed operation first uses a point convolution to scale channels of the input feature map, and then, the output feature map is flattened according to the dimension size. Note that the bilinear pooling operation is performed on the same pixel location of the channel. The extracted feature map is used for multi-layer scaling to obtain the weight vector with the same dimension to the original feature map, and the weight vector reflects the interaction among all channels. Finally, the input feature will be multiplied with the weight vector to form the feature weighted by the attention.

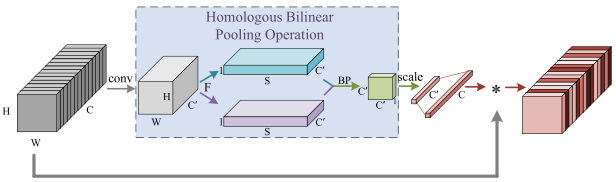


Fig. 2 The proposed bilinear pooling attention module (BPAttLayer)

Encoding scheme

The encoding scheme of neural architecture is an important issue concerned in ENAS, directly reflecting the search space in evolutionary process. Previous works [16,27,32,36] proposed various encoding strategies on network architectures, such as fixed-length topological encoding, variable length hyperparameter coding, etc. In this paper, we propose an unified encoding scheme combining the topological structure and computing operation together. Therefore, the dimension of decision vector is increased by a large amount and the search space is enlarged greatly in the evolutionary process. In the meanwhile, this unified encoding scheme can also help explore the relationship between topological structure and computing operation, relieving the drawback of separate encoding.

As mentioned above, the neural network architecture is composed of two basic cells: the normal cell and the reduction

cell. Accordingly, we represent the individual as a quadruple $E = (k_1, NC, k_2, RC)$, where NC and RC denote the encoding expression of the two cell structures, respectively, and k_1 and k_2 represent their nodes number. The directed acyclic graph is utilized to represent the cell structure. Taking Fig. 1b as an illustrative example, the normal cell and the reduction cell both have two inputs (denoted as $h[-1]$ and $h[0]$), and 6 nodes and 4 nodes, respectively.

Furthermore, a tuple composed of two vectors is utilized to encode the directed acyclic graph, which includes topological structure and node operation types. For example, the encoding of normal cell structure $NC = (Link, Ops)$, where $Link$ denotes a vector mapping the connection relationship among nodes, and its length is $\frac{(k_1+3)\times k_1}{2}$, and Ops represents a vector of length k_1 that denotes the operation index corresponding to each node. Therefore, each node could be divided into two parts

$$node_i = (Link_i, Ops_i),\ i \in \{1, 2, ..., k_1\}, \tag{3}$$

where $Link_i$ is a binary vector of length $i + 1$, denoting the connection relationship between $node_i$ and all precursor nodes, and Ops_i is the operator index of $node_i$. It has been mentioned in Table 1, and the proposed search space contains 11 operators, so $Ops_i \in \{0, 1, \ldots, 10\}$. The encoding of reduction cell RC is similar to NC. For a better understanding, we take the fourth node of normal cell in Fig. 1b as a example

$$node_4 = (Link_4, Ops_4) = (1, 0, 1, 0, 0, 5), \tag{4}$$

where the binary vector $Link_4 = (1, 0, 1, 0, 0)$ indicates that $node_4$ only links to the first input node $h[-1]$ and $node_1$, and the sub-vector $Ops_4 = 5$ means that $node_4$ adopts the convolution with kernel size 3x3 (i.e., operator **Conv 1*7+7*1**). Different colors are utilized to distinguish node operations in Fig. 1.

The proposed AEMONAS

Overall framework

In this section, the implementation details of the proposed AEMONAS will be described. First, we will outline the the framework of our proposed AEMONAS. Afterward, we will elaborate on the two crucial components in AEMONAS: (1) The architecture entropy-based sampling strategy, which can make the distribution of population more uniform. (2) The genetic operators, which are tailored for the unified encoding of topological structure and computing operation.

Overall framework

For NAS algorithms, not only the recognition performance, but also the deployment convenience should be taken into account. Different applications have different requirements on the performance and complexity of the neural network, and it is necessary to trade off these two factors according to the deployment scenarios. In this work, to ensure a balance between performance and resource constraints, the accuracy and the complexity are regarded as two objectives to be optimized simultaneously, while NSGA-II algorithm [5] is taken as the optimizer due to its excellent stability and competitiveness in solving multi-objective optimization problems. Compared with the existing method of multi-objective ENAS, the proposed AEMONAS adopts the sampling strategy based on architecture entropy in the process of population evolution to maintain the even diversity of the architectures, and the novel genetic operators are designed to tailor for the unified encoding scheme and accelerate the convergence. The overall framework of AEMONAS is shown in Fig. 3.

Algorithm 1: Framework of AEMONAS

Input: Generation number G, Population size N, Node number of normal cell k_1, Node number of reduction cell k_2, Training data D_{trn}, Validation data D_{vld}, The number of candidate architectures K, Epoch number Ep;
Output: Top-K candidate neural architectures $Arcs$;
1: $P_0 \leftarrow$ Initialize a population $\{(k_1, NC, k_2, RC)\}^N$
2: Evaluate individuals in P_0 by training the decoded architectures for Ep epochs on D_{trn} and validate the accuracy on D_{vld}
3: **for** $t \leftarrow 0$ to G-1 **do**
4: // mating pool selection
5: $P'_t \leftarrow$ Select N parents from P_t
6: $Q_t \leftarrow$ Sampling offspring population by Algorithm 2
7: Evaluate individuals in $P_t \cup Q_t$ by training the decoded architectures for Ep epochs on D_{trn} and validate the accuracy on D_{vld}
8: // environment selection
9: $P_{t+1} \leftarrow$ Select N individuals by non-dominated sort
10: **end for**
11: $Arcs \leftarrow$ Select top-K individuals by non-dominated sort from P_G
12: **return** $Arcs$

First, an initial population P_0 containing N randomly produced individuals is generated. Then, all individuals in the initial population are evaluated according to their performance, which refers to decoding each individual into corresponding convolutional neural architectures and training them via stochastic gradient descent method for a certain number of epochs (the purple part in Fig. 3. Afterward, competitive parent individuals are selected from current population P_t to form the mating pool P'_t through binary

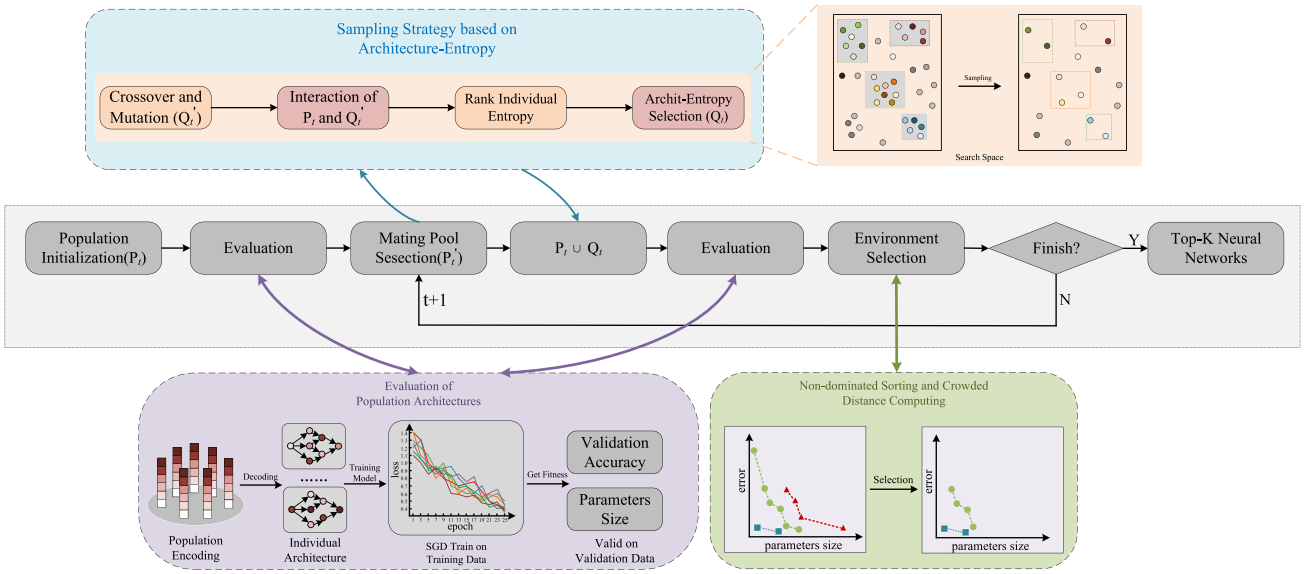


Fig. 3 The overall framework of the proposed AEMONAS

tournament selection strategy. An alternative population Q'_t consisting of M new individuals is immediately produced by performing the novel crossover and mutation operations (depicted in the next section) on the mating pool P'_t, where the value of M is much greater than N. Then, for each individual Q_i in Q'_t, the architecture entropy of P_t before and after Q_i is inserted into P_t is calculated, and the change which is taken as the sampling metric is assigned to Q_i. Subsequently, the proposed sampling strategy based on architecture entropy is utilized select top N offspring individuals with a wide range of diversity to produce offspring population Q_t. In environment selection, P_t and Q_t are merged into a combined population with the size of $2N$, and all individuals are trained to update the fitness values. Followingly, N individuals are selected as the parent population of the next generation P_{t+1} according to the non-dominated ranking criteria. The above steps will repeat until the termination condition is satisfied. Finally, AEMONAS outputs top K excellent architectures, so that different high-performance neural networks are selected for different resource constraints. Algorithm 1 presents more details of the proposed AEMONAS.

Architecture entropy based sampling strategy

In traditional multi-objective evolutionary optimization problems, population diversity is significant to avoid premature convergence. Premature convergence refers to that the population decays into a collection of individuals with highly similar genotypes before the search space is fully explored, and it is an obvious problem in ENAS. To address this issue, we define an indicator inspired by population entropy [37,48], named **architecture entropy**, to evaluate the architecture diversity in the population and design a corresponding sampling strategy to retain the architectures with the potential to enhance population diversity in the stage of offspring generation.

Architecture entropy indicator

To quantify the architecture diversity of a population, we first carefully design the metric of architecture entropy. First, each individual i in the population P_t is assigned with a density value D_i, which represents the density of connections among all the nodes of the cell

$$\begin{cases} D_i = \dfrac{k_1|NL| + k_2|RL|}{k_1 + k_2}, \\ NL = \{e | e \in NC.Link \cap e = 1\}, \\ RL = \{e | e \in RC.Link \cap e = 1\}, \end{cases} \tag{5}$$

where $|\cdot|$ means the cardinality of a set, and NL and RL denote the sets of binary code with the value of 1 in link encoding in normal cell and reduction cell, respectively. Then, the distribution of D value is evenly divided into Q intervals according to its scope:

$$\begin{cases} S_1, S_2, ..., S_Q, \ Q \leq N \\ Q = \frac{D_{max} - D_{min}}{\theta}, \\ D_{max} = \frac{k_1^2(k_1+3)+k_2^2(k_2+3)}{k_1+k_2} \\ D_{min} = \frac{k_1^2+k_2^2}{k_1+k_2}. \end{cases} \tag{6}$$

In this equation, $S_q (1 \leq q \leq Q)$ denotes the q-th interval, and Q is the number of intervals, with θ indicating the granu-

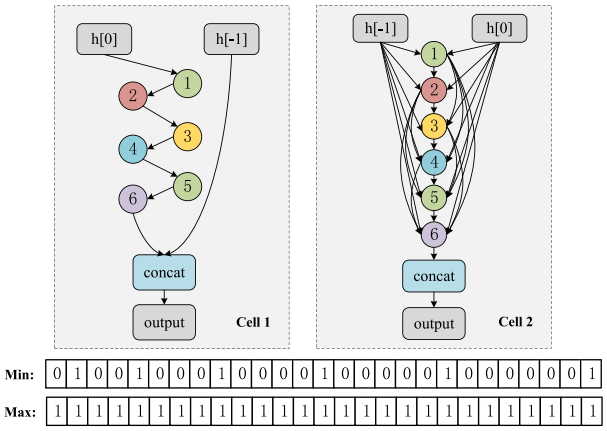


Fig. 4 Example of two cell structures with extreme density values. The former indicates the most sparse structure, while the latter represents the most densely connected structure

larity (the range of D values of each interval), and D_{max} and D_{min} denote the maximum and minimum D values of the individual. Note that, as shown in Fig. 4, the minimum corresponds to the cell structure where each node has only one precursor node, while the maximum corresponds to the cell structure where each node connects to all precursor nodes. Subsequently, the architecture entropy of P_t is calculated according to the D values of N individuals by counting the number of individuals in each interval ($|S_q|$)

$$\begin{cases} AE = -\sum_{q=1}^{Q} p_q log(p_q), \\ p_q = \frac{|S_q|}{N}, \ q \in \{1, 2, ..., Q\}. \end{cases} \tag{7}$$

where p_q denotes the ratio of the number (S_q) of the individuals populated in the interval where current individual is located in to the population size N, and AE denotes the architecture entropy of the population.

Sampling strategy

According to the definition in Eq. 7, when the density values of all individuals in the population are the same, they will all be distributed in an interval, and the architecture entropy value of P_t reaches the minimum, $AE = 0$. When the density values of all individuals in a population are more evenly distributed, the architecture entropy value will be much higher. Based on this observation, we propose a sampling strategy to select individuals with the potential to enhance population diversity during offspring generation so as to explore the search space more effectively.

The proposed architecture entropy-based sampling strategy mainly consists of four steps: First, the crossover operation and mutation operation are performed on the individuals in the mating pool to generate M new offspring individuals. Note that M is much larger than the population size N, and it is necessary to filter out some individuals for environmental selection. Then, as illustrated in Fig. 5, the candidate offspring population and the parent population P_t interact. Assuming the architecture entropy of P_t is equal to AE_1, for each candidate offspring, it is successively added to the parent population P_t, causing the the architecture entropy of P_t become AE_2. The change $(AE_2 - AE_1)$ is calculated as the selection metric of current offspring. The larger the metric is, the greater the contribution potential of this individual to maintain the population diversity will be expected. Subsequently, the selection metrics of all candidate offsprings are ranked in the descending order. Finally, according to the sorted individual index, the first N individ-

Algorithm 2: Architecture Entropy Based Sampling Strategy

Input: Parent population P_t, Mating pool P'_t, Sampling number M, Interval granularity θ, Probability of crossover α, Probability of mutation β;
Output: Offspring population Q_t;

$Q'_t \leftarrow \emptyset$ // initialize a offspring sample set.
$Q_t \leftarrow \emptyset$ // initialize the offspring population.
while $|Q'_t| < M$ **do**
  $p_1, p_1 \leftarrow$ Randomly select two individuals from P'_t
  // performing crossover operation
  $\rho \leftarrow$ Randomly generate a number from (0,1]
  if $\rho < \alpha$ **then**
    if rand() < 0.5 **then**
      $q_1, q_2 \leftarrow$ HolisticCrossover(p_1, p_2)
    else
      $q_1, q_2 \leftarrow$ HybridCrossover(p_1, p_2)
    end if
  else
    $q_1, q_2 \leftarrow p_1, p_2$
  end if
  // performing mutation operation
  $\rho \leftarrow$ Randomly generate a number from (0,1]
  if $\rho < \beta$ **then**
    // based on probability-level
    $q_1, q_2 \leftarrow$ Mutation(q_1, q_2)
  end if
  $Q'_t \leftarrow Q'_t \cup q_1 \cup q_2$
end while
$E_{ori}, E_{Arr} \leftarrow AE(P_t, \theta), \emptyset$
for $i \leftarrow 1$ to M **do**
  $E \leftarrow AE(P_t \cup Q'_t[i], \theta) - E_{ori}$
  $E_{Arr}[i] \leftarrow \{E, i\}$
end for
for $i \leftarrow 1$ to M **do**
  $t \leftarrow$ Choose the item with max AE value from E_{Arr}
  $E, idx \leftarrow t[0], t[1]$
  $Q_t \leftarrow Q_t \cup Q'_t[idx]$
  $E_{Arr} \leftarrow E_{Arr} - t$
end for
$Q_t \leftarrow Q_t[: |P_t|]$
return Q_t

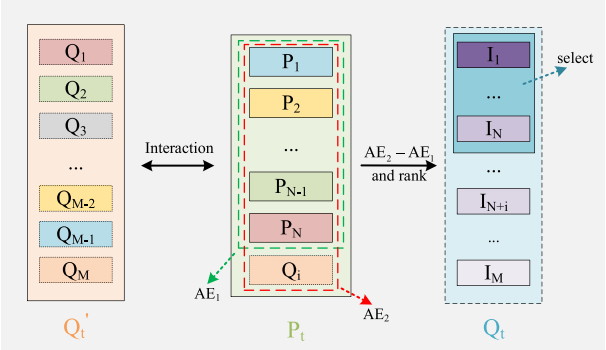


Fig. 5 Example of the proposed architecture entropy-based sampling strategy

uals are selected from the candidate offspring population, which has the same population size with the parent population, and hence, the final offspring population and the parent population are merged together for further environmental selection. The architecture entropy-based sampling process is shown in Fig. 5.

The computational complexity of this process is $O(M * logM)$, which does not require additional training for individuals. Compared with the training cost of the model, the time-consuming of this strategy is minimal. The details of the proposed sampling strategy are described in Algorithm 2.

Genetic operator

To promote the convergence of the proposed AEMONAS algorithm, we design an effective genetic operator for the offspring generation tailored for the unified encoding scheme of topological structure and computing operation. Two different crossover modes are suggested to balance the coupling and interdependency between topological structure and computing operation.

Crossover

As our architecture encoding strategy considers both topological information and computing operation, based on this characteristic, the proposed crossover operator includes two modes: holistic crossover and hybrid crossover. Holistic crossover refers to regarding connection information and computing operation of each node as a whole in the process of crossover. An intuition of this approach lies on the fact that the excellent genes of the parents may be the operation and the corresponding topological relationship simultaneously, and so we could preserve their overall information. On the contrary, hybrid crossover attempts to break the coupling of operation type and connection relation, and makes the connection encoding and operation encoding in the parent individuals crossover separately, and so, the excellent topological gene and operation gene of the parents could be intuitively inherited at the same time. To maintain a balance between the two crossover operation, the same selection probability for both crossover types is adopted.

Figure 6 shows the details of the proposed crossover operator, including holistic crossover and hybrid crossover. For convenience, we use the operations of normal cells as examples. In fact, reduction cells will also perform the same operations to ensure uniformity. N_1 and N_2 represent normal cells in two parent individuals, respectively, which are both composed of topology encoding $Link$ and the operator encoding Ops. In holistic crossover, a randomly generated breakpoint is positioned at the junction of $node_4$ and $node_5$, so that $node_5$ and $node_6$ in N_1 and N_2 are swapped as a whole. In hybrid crossover, the topology encoding $Link$

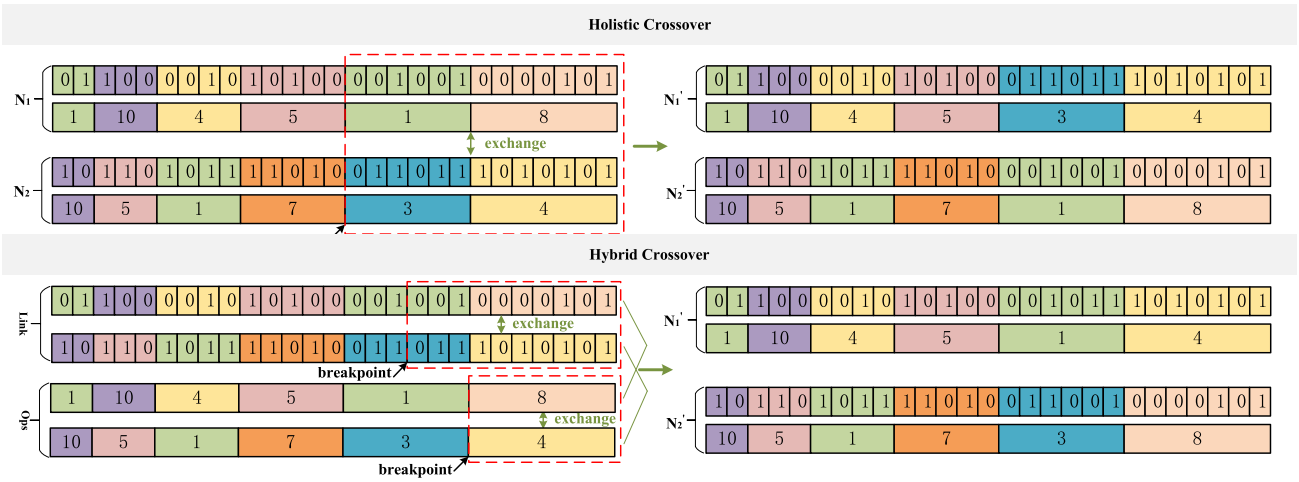


Fig. 6 The proposed crossover operator. Illustrative example of holistic crossover and hybrid crossover between normal cells N_1 and N_2 in two parent individuals

and the operator encoding Ops are listed together and both perform the single-point crossover at randomly generated breakpoints.

Mutation

The mutation operator plays a crucial role for escaping locally optimal space in traditional evolutionary algorithms. In this paper, a single-point mutation operator is adopted. The connections and operations are mutated separately with different probability. Note that the target operation type to be mutated to is not completely random, but based on the probability calculated through counting occurrence of each operation in the parent population

$$P(Op_i) = \frac{exp(N(Op_i))}{\sum_{o \in O} exp(N(o))}, \ i \in \{1, 2, ..., |O|\}, \tag{8}$$

where $P(Op_i)$ denotes the probability that operator Op_i is selected as the target operation for current node, and $N(Op_i)$ denotes the number of occurrences of operator Op_i in parent population; $|O|$ is the number of predefined operators in the search space ($|O| = 11$ in this paper).

As shown in Fig. 7, the single-point mutation is performed in the topology encoding $Link$ and the operator encoding Ops of the normal cell N_1, respectively. In the $Link$, the first bit in $node_3$ is randomly selected and the corresponding code 0 is mutated to 1. Similarly, in the Ops, the operation code of $node_4$ is randomly selected, and the original code 5 mutates to 3 (i.e., from Conv 1*7+7*1 to DSConv 3*3), while the target code 3 is not randomly generated, but is obtained by calculating the frequency of occurrence of each operation in the parent population.

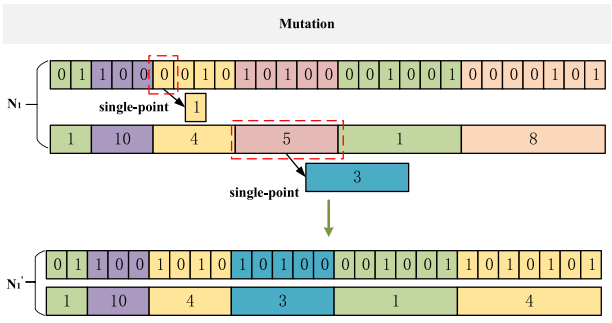


Fig. 7 Illustrative example of the proposed probability-level-based mutation operator of N_1

Empirical studies

Experimental settings

Benchmark datasets

In the experiments, the benchmark datasets CIFAR10 and CIFAR100 [12] are adopted to evaluate the performance of the proposed AEMONAS and the comparison methods, and these benchmark datasets are widely used in the state-of-the-art NAS algorithms.

CIFAR-10 is a 10-category RGB color images dataset, which contains 60000 images with the size of 32*32. The categories include cars, aircraft, cats and dogs, etc. CIFAR-100 [12] is an advanced version of CIFAR-10 and has the same format with CIFAR-10. CIFAR-100 also includes 60000 32x32 images, but the category number is 100. Note that each dataset is divided into training set and testing set, which consist of 50000 images and 10000 images, respectively.

To address the lack of training data, we adopt the standardized augmentation strategy: cut-out operation and random horizontal flip operation. All the methods in the experiments are trained and tested on the same datasets.

Comparison methods

In the experiments, we investigate the competitiveness of the proposed AEMONAS by comparing it with the state-of-the-art NAS algorithms. According to the reviews in the literature, the peer competitors selected could be divided into three categories according to search strategies: manually designed methods, non-EA-based methods (e.g., reinforcement learning-based, gradient-based, etc.) and EA-based methods. The manually designed architectures include the classic models: ResNet [8], Wide ResNet [43], DenseNet [11], SENet [10] with attention mechanism, as well as lightweight networks such as MobileNetV2 [29] and ShuffleNet [47]. Non-EA-based NAS algorithms mainly include the reinforcement learning-based and gradient strategy-based approaches: PNAS [15], NASNet [53], MetaQNN [2], DARTS [17], Block-QNN-S [49], ENAS [25], and NAO [21]. EA-based NAS approaches include Genetic-CNN [40], NSGANet [19], AmoebaNet [27], AE-CNN [31], Hierarchical Evolution [16], SI-EvoNet [46], and CARS-E [42].

Parameter settings

The determination process for ultimate neural network model includes two stages: the model search stage and the top K

optimal model retraining stage. The parameter settings of the proposed AEMONAS in these two stages are set as follows:

Search stage: In the search stage, the population size N and the number of evolutionary iterations G are set to 20 and 25, respectively, and all the generated architectures have 5 cells ($S = 1$) and the number of channels of nodes in each cell is set to 16. To ensure the learning ability of the overall network, all architectures are kept the same with normal cell containing 8 nodes, and reduction cell containing 6 nodes ($k_1 = 8, k_2 = 6$). The individual evaluation process is completed by training 25 epochs ($Ep = 25$) using the SGD optimizer. In the training of neural network of the search stage, the initial learning rate and momentum rate of the optimizer are set to 0.1 and 0.9, respectively. The cosine annealing learning strategy is adopted to update the learning rate with the iterative training. To avoid overfitting, L2 weight decay strategy with a value of 1×10^{-4} is adopted. The batch size of training set and validation set is configured to 128 and 512, respectively. For the architecture entropy-based sampling, M is set to 100 (much larger than N), and the interval granularity θ is fixed to 4. For all the comparison NAS algorithms, the parameters for training the neural networks are kept the same. For EA-based methods, the population size and evaluation times are set the same.

Retraining stage: After the search stage, only the basic network architecture is determined. To fully exploit the potential of the network architecture, the adjustment on the network architecture and retraining should be undertaken. Among the pareto optimal individuals obtained after the search stage, the top 2 architectures of different complexity are selected as the final neural networks for retraining.

During the retraining stage, the stacking time S of the normal cell is extended to 6, the number of channels in the initial cell is set to 32, and the channel will be doubled as the spatial resolution of the reduction cell decreases. The architecture is trained with 600 epochs by the SGD optimizer with momentum. The corresponding parameters are set the same with those in the search stage. Distinct from the search stage, a cosine annealing learning rate mechanism with an initial learning rate of 0.025 is utilized. The L2 weight decay rate is set to 5×10^{-4}. In addition, to regularize the network training, the dropout strategy is adopted, while each path will be dropped with a proportion of 0.2.

Note that for both search stage and retraining stage, training set is divided into training set and validation set with the ratio of 9:1. The performance of each network architecture is evaluated on the validation set. Testing set is utilized for the final performance evaluation of the architecture, and is not allowed to be used to guide search and training. All experiments including search phase and retraining phase are performed on two Nvidia GeForce GTX 1080Ti. The source code of the proposed AEMONAS is available at https://github.com/LabyrinthineLeo/AEMONAS.

Competitiveness of the proposed AEMONAS

The ultimate performances of different NAS methods are evaluated under the metrics of top-1 accuracy on the testing set, the parameter volume of the output neural network, and the search cost (GPU days). Top-1 accuracy denotes the test classification accuracy, and parameter volume can reflect the complexity of the output neural network. Obviously, the search cost denotes the computational complexity of the NAS algorithms.

The detailed experimental results of the proposed AEMONAS and the comparison methods on CIFAR-10 are illustrated in Table 2. At the end of the search process on the CIFAR-10 dataset, two architectures are selected from the pareto solution set searched by the proposed AEMONAS and adjusted in the retraining stage, and are named as AEMONet-

Table 2 The comparison of the proposed AEMONAS and existing peer competitors in terms of Top-1 accuracy and consumed computational cost on the CIFAR-10 datasets

Architecture	Top-1 Acc (%)	Params (M)	Search cost (GPU days)	Search method
Wide ResNet ($k = 2$) [43]	94.67	2.2	–	Manual
Wide ResNet ($k = 4$) [43]	95.03	8.9	–	Manual
DenseNet ($k = 12$) [11]	95.90	7.0	–	Manual
DenseNet ($k = 24$) [11]	96.26	27.2	–	Manual
SENet [10]	95.95	11.2	–	Manual
MobileNetV2 [29]	94.56	2.1	-	Manual
ShuffleNet [47]	90.87	1.06	-	Manual
PNAS [15]	96.59	3.2	225	SMBO
NASNet-A [53]	97.35	3.3	2000	RL

Table 2 continued

Architecture	Top-1 Acc (%)	Params (M)	Search cost (GPU days)	Search method
Block-QNN-S [49]	95.62	6.1	90	RL
MetaQNN [2]	93.08	11.2	80	RL
ENAS+Cutout [25]	97.11	4.6	0.5	RL
DARTS(first order) [17]	97.00	3.3	1.5	Gradient based
DARTS(second order) [17]	97.24	3.3	4.0	Gradient based
NAO [21]	96.82	10.6	200	Gradient based
Genetic-CNN [40]	92.90	−−	17	Evolution
NSGANet [19]	97.25	3.3	4	Evolution
AmoebaNet-A [27]	96.66	3.2	3150	Evolution
AE-CNN [31]	95.70	2.0	27	Evolution
Hierarchical Evolution [16]	96.37	15.7	300	Evolution
SI-EvoNet [46]	96.02	0.51	0.458	Evolution
CARS-E [42]	97.14	3.0	0.4	Evolution
AEMONet-A	**96.88**	**1.20**	**3.4**	Evolution
AEMONet-B	**97.30**	**2.23**	**3.4**	Evolution

† – means that the corresponding result was not published
Bold highlights the performance effects of our algorithm and the corresponding improvement ratio

A and AEMONet-B (as shown in Fig. 8), while the amount of their parameters is equal to 1.20M and 2.23M, respectively. From Table 2, it could be seen that our AEMONet-A and AEMONet-B achieved 96.88% and 97.30% classification accuracy, respectively. Compared with the manually designed models, both of AEMONet-A and AEMONet-B have higher accuracy and fewer parameters. Especially, in comparison with the commonly used DenseNet(k=24), our network architectures have ten times fewer parameters without any loss of accuracy. Compared with the 8 non-EA based NAS algorithms, our AEMONet-B outperforms PNAS, Block-QNN-S, MetaQNN, ENAS, DARTS, and NAO, and only a little worse than NASNet-A by (0.05%), but the number of parameters and the search cost are dramatically decreased. Compared with the 7 EA-based NAS algorithms, AEMONet-B achieves better performance than all of them. Especially, in comparison with AE-CNN, the parameter volumes are almost identical, but AEMONet-B can achieve a 1.6% improvement in terms of classification accuracy. In the meanwhile, AEMONet-B can demonstrate an obvious superiority over the classical NSGANet with a parameter reduction by two-thirds. AEMONet-A with lower model complexity is also very competitive with architectures with similar lightweight architectures (such as ShuffleNet, SI-EvoNet, etc.).

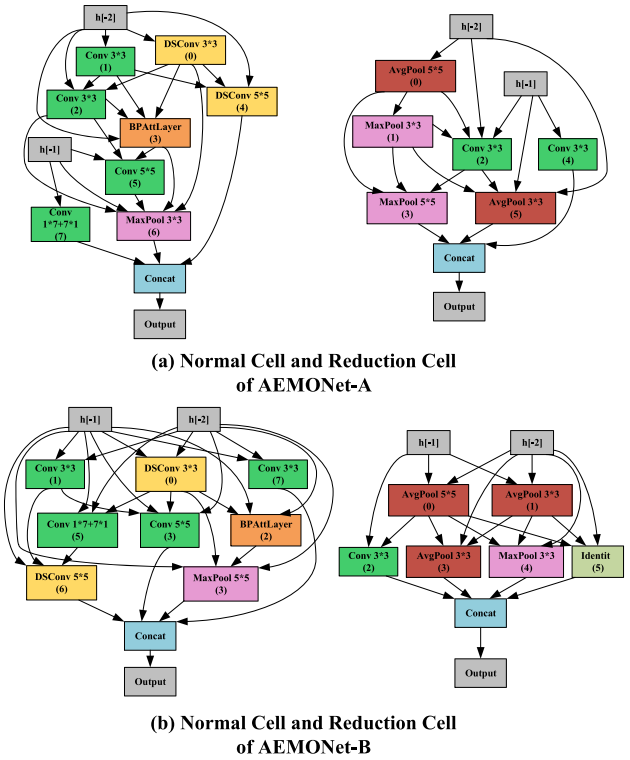


Fig. 8 Details of normal cell and reduction cell of the AEMONet-A and AEMONet-B achieved by the proposed AEMONAS on CIFAR datasets

To further investigate the competitiveness of the proposed AEMONAS, we transfer AEMONet-A and AEMONet-B searched on CIFAR-10 to CIFAR-100. The detailed experimental results are presented in Table 3. From Table 3, it could be observed that the transferred AEMONet-B achieves a 83.01% test classification accuracy on CIFAR-100. This accuracy is better than 14 comparison algorithms but worse than 3 comparison algorithms (NASNet, NAO, and Amoe-

Table 3 The comparison of the proposed AEMONAS and existing peer competitors in terms of Top-1 accuracy and consumed computational cost on the CIFAR-100 datasets

Architecture	Top-1 Acc (%)	Params (M)	Search cost (GPU days)	Search method
Wide ResNet ($k = 10$) [43]	79.50	36.5	–	Manual
DenseNet-BC ($k = 40$) [11]	82.82	25.6	–	Manual
SENet ($k = 4$) [10]	76.15	−−	–	Manual
MobileNetV2 [29]	77.09	2.1	–	Manual
ShuffleNet [47]	77.13	1.06	–	Manual
PNAS [15]	80.47	3.2	225	SMBO
NASNet-A [53]	83.42	3.3	2000	RL
Block-QNN-S [49]	82.95	6.1	90	RL
MetaQNN [2]	72.86	11.2	80	RL
ENAS [25]	82.73	4.6	0.5	RL
DARTS [17]	82.46	3.3	4	Gradient based
NAO [21]	84.33	10.8	200	Gradient based
Genetic-CNN [40]	70.97	−−	17	Evolution
NSGANet [19]	79.26	3.3	8	Evolution
AmoebaNet [27]	84.20	3.2	3150	Evolution
AE-CNN [31]	79.15	5.4	36	Evolution
SI-EvoNet [46]	79.16	0.99	0.813	Evolution
AEMONet-A	**79.90**	**1.21**	**3.4**	Evolution
AEMONet-B	**83.01**	**2.24**	**3.4**	Evolution

† **-** means that the corresponding result was not published
Bold highlights the performance effects of our algorithm and the corresponding improvement ratio

baNet). However, their search cost is much higher than that of our AEMONAS, and especially, AmoebaNet even reaches 3150 GPU days, which cannot be afford by common users. Moreover, the architectures achieved by the comparison methods usually have a larger number of parameters (e.g., NAO even reaches 10.8M). Meanwhile, the transferred AEMONet-A also achieves a classification accuracy of 79.90%. Therefore, the proposed AEMONAS performs better or equivalent compared with state-of-the-art NAS algorithms on the CIFAR-100 dataset.

In summary, under each metric, our proposed AEMONAS can achieve the best or sub-optimal performance. If all the metrics are considered by the whole, our proposed AEMONAS obviously outperforms the comparison methods.

Effectiveness of the architecture entropy-based sampling strategy

In this section, we will verify the effectiveness of the architecture entropy-based sampling strategy in the proposed AEMONAS. For a fair comparison, we first build a baseline ENAS framework called naive MO-ENAS, which differs from the proposed AEMONAS only in that it uses the original NSGAII algorithm directly, and the architecture entropy-based sampling is removed from the AEMONAS framework. We then perform naive MO-ENAS to complete the search process on CIFAR-10 with the same parameter settings as described earlier.

To measure the diversity of the population, we investigate the range of density values and the variational tendency of architecture entropy during the search stage of naïve MO-ENAS and AEMONAS on CIFAR-10, respectively. As shown in Fig. 9, the X-axis represents the index of the iteration generation, while Y-axis denotes the density values (left) and the architecture entropy (right), and the population is sampled at the interval of six generations to execute a evaluation. It could be obviously observed that in the search process, the range of individual density values of the population will gradually decrease, but the magnitude of reduction of AEMONAS is significantly smaller than that of naive MO-ENAS. Meanwhile, in terms of architecture entropy values, the declining trend of AEMONAS is slower than that of naive MO-ENAS.

At the same time, we record the performance of all architectures in each generation population during the search stage. As shown in Fig. 10, the green and purple curves represent the changes in average accuracy of all architectures of AEMONAS and naive MO-ENAS in each generation, respectively, and the light-colored part represents the upper

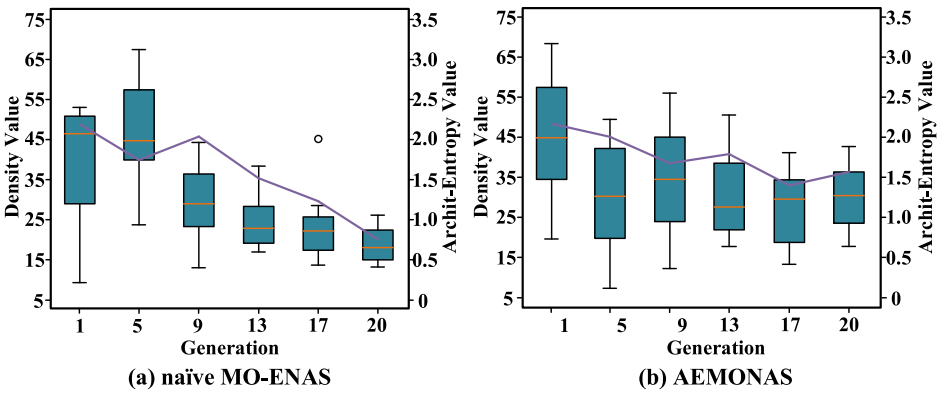


Fig. 9 Boxplot of the range of density values of all architectures in each generation population and curve change of architecture entropy values, where six generations of equal intervals are taken as an example. **a** and **b** are the diversity results of naive MO-ENAS and AEMONAS, respectively

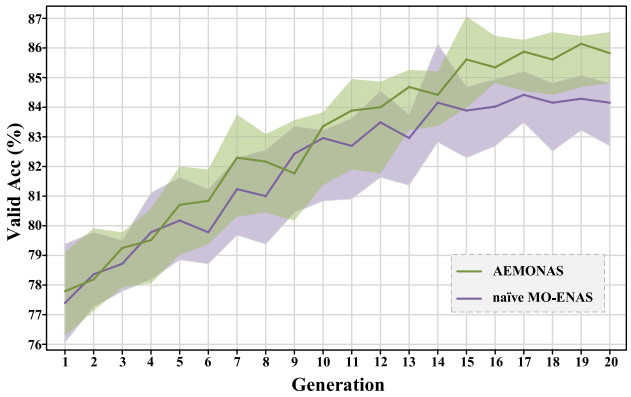


Fig. 10 Curve change of validation average accuracy of all architectures in each generation of population during the search process of AEMONAS and naive MO-ENAS, and the light-colored part represents the upper and lower limits of the validation accuracy

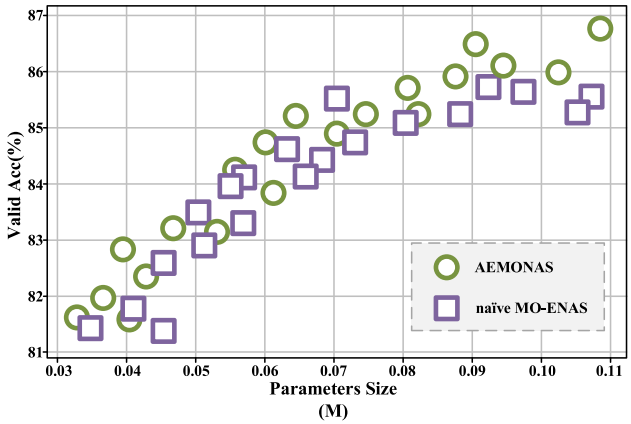


Fig. 11 Example of the number of parameters and validation accuracy of the final architectures searched by AEMONAS and naive MO-ENAS

and lower limits. It can be clearly found that AEMONAS has better exploration ability than naive MO-ENAS. Finally, to intuitively compare the results explored by both of them, Fig. 11 shows the number of parameters and validation accuracy of the final architectures obtained by AEMONAS and naive MO-ENAS. It can be seen from the figure that the architectures searched by the proposed AEMONAS are better.

To summarize, the proposed sampling strategy based on architecture entropy can effectively maintain the diversity of the population, alleviate the problem of premature convergence, and significantly improve the exploration ability of evolutionary neural architecture search algorithms.

Ablation studies

In this section, we construct two ablation experiments to further verify the effectiveness of the proposed architecture search method and the designed attention mechanism operator, respectively.

First, considering that some NAS methods in the peer competitors do not use the attention mechanism, we attempt to add our proposed BPAttLayer operator to their search space, and then compare the corresponding search results to further verify the comprehensive availability of our AEMONAS. Due to the constraints of computing resources and reproducibility of code, we select three neural architecture search algorithms including DARTS [17], NSGANet [19], and SI-EvoNAS [46] as experimental comparison methods. The designed BPAttLayer module has significant flexibility and adaptability, which can be easily embedded into the search space of the three NAS methods. In addition, the experimental parameters set in the search stage and retraining stage are consistent with those described in the corresponding papers.

Table 4 shows the performance on CIFAR10 and CIFAR100 of the optimal neural networks finally searched by the three NAS methods after the addition of BPAttLayer in the search space. The values in brackets represent the classification accuracy changes compared with the original algorithm. It can be found that in addition to the slight performance decline of NSGANet on CIFAR-10, the performances of other algorithms are improved to some extent. However, our search algorithm still has a clear advantage, which also proves the overall validity of the proposed AEMONAS.

Table 4 The classification accuracy(%) of DARTS+BPAttLayer, NSGANet+BPAttLayer, SI-EvoNAS+BPAttLayer, and our AEMONAS on CIFAR-10 and CIFAR-100 datasets

Algorithms	Test accuracy (%)	
	CIFAR-10	CIFAR-100
DARTS+BPAttLayer	97.16(**+0.16**)	82.57(**+0.11**)
NSGANet+BPAttLayer	97.21(**–0.04**)	80.97(**+1.71**)
SI-EvoNAS+BPAttLayer	96.65(**+0.63**)	80.59(**+1.43**)
AEMONAS	**97.30**	**83.01**

Bold highlights the performance effects of our algorithm and the corresponding improvement ratio

Table 5 The results of the ablation experiment on BPAttLayer. Each row denotes the result of the same dataset, and each column represents the result of replacing BPAttLayer in AEMONet-A and AEMONet-B

Dataset	Test accuracy (%)			
	BPAttLayer	Identity	Conv 3*3	**DSConv 3*3**
CIFAR-10	**96.88**	94.89	95.37	95.95
	97.30	95.08	95.87	96.22
CIFAR-100	**79.90**	77.35	77.83	78.18
	83.01	80.14	81.31	81.09

† **DSConv** denotes the depth separable convolution
Bold highlights the performance effects of our algorithm and the corresponding improvement ratio

In addition, to investigate the adaptability of the designed attention module (i.e., BPAttLayer operation) in the search space and its contribution to improving the expression ability of the neural architecture, we perform ablation experiments by replacing the BPAttLayer in the complete architecture with ordinary operations. First, identity operation, 3*3 normal convolution operation, and 3*3 depth separable convolution are utilized to replace the BPAttLayer in the two neural networks (i.e., AEMONet-A and AEMONet-B), respectively, and then, the changed architectures are trained on CIFAR-10 and CIFAR-100 datasets. For fair comparison, all training parameters are consistent with those described above.Table 5 represents the performance results and comparisons after replacing BPAttLayer with different operations on three datasets. It can be observed that the classification accuracy of all the replaced architectures decreases to varying degrees. In particular, the classification accuracy of AEMONet-B (Identity) decreases by 2.22% and 2.77%, respectively, on CIFAR-10 and CIFAR-100. In addition, the performances of AEMONet-B (Conv 3*3) and AEMonet-B (DSConv 3*3) on CIFAR datasets also decrease significantly.

Application to osteoporosis diagnosis

Osteoporosis is one of the most common chronic metabolic bone diseases, characterized by reduced bone mass, destruction of bone tissue microstructure, and increased bone fragility [45]. Due to the wide prevalence, the early diagnosis is quite essential to reduce the risk of osteoporotic fractures, and the identification and diagnosis based on X-ray image is one of the most convenient and effective methods to screen potentially susceptible population. Recently, many studies focus on the diagnosis of osteoporosis through the recognition and classification of medical X-ray images. The convolutional neural network (CNN) is utilized to automatically extract features, which are more effective for classification [13]. However, how to design an appropriate network architecture is an challenging issue to researchers. In this section, we will investigate the performance of AEMONAS in solving this problem.

We construct a lumbar spine X-ray image dataset from real medical scenarios for osteoporosis diagnosis. The lumbar spine dataset is a three-category lumbar X-ray images dataset, and includes 1531 high-resolution images which containing the first to the fourth lumbar vertebrae (L1–L4). Each image is labeled as one of the three categories: normal, slight, and serious, indicating normal bone mass, osteopenia and osteoporosis, respectively. This dataset contains two subsets, the anteroposterior view (PA) and the lateral view (LAT), which consist of 747 and 784 images, respectively (as shown in the Fig. 12). Due to the large difference of images from these two perspectives, we divide them into two classification tasks. The AEMONAS is utilized to search the best neural architecture for this image classification problem. All images

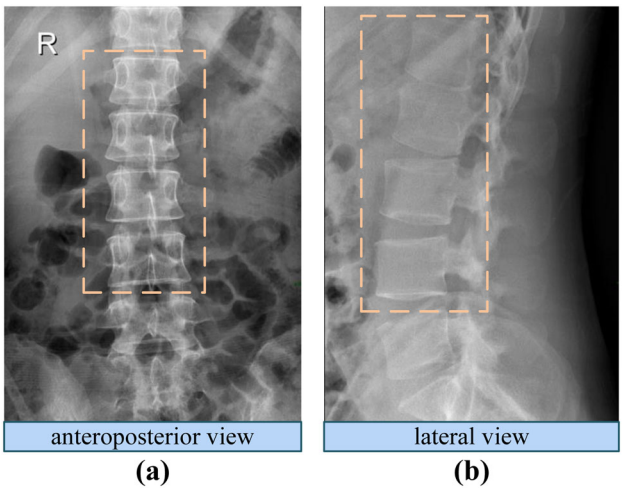


Fig. 12 Example of two lumbar spine X-ray images; the dotted box represents the first to the fourth lumbar vertebrae from top to bottom (L1–L4): **a** anteroposterior view and **b** lateral view

Table 6 The comparison among the proposed AEMONAS and selected peer competitors in terms of test accuracy (%) and Test AUC (%) on Lumber Spine Dataset of Anteroposterior view (PA) and Lateral view (LAT)

Architecture	Params (M)	PA Test Acc (%)	PA Test AUC (%)	LAT Test Acc (%)	LAT Test AUC (%)	Type
ResNet-34 [8]	21.8	84.810	90.431	86.228	95.064	Manual
Wide ResNet (k = 4) [43]	8.9	86.709	92.693	85.629	95.318	Manual
DenseNet (k = 12) [11]	7.0	88.358	93.791	90.216	96.024	Manual
SENet (k = 4) [10]	11.2	86.076	92.501	87.452	93.284	Manual
NASNet-A [53]	3.3	88.241	93.449	89.419	92.342	RL
ENAS [24]	4.6	89.405	94.238	89.222	94.573	RL
DARTS [17]	3.3	86.709	92.534	90.401	96.671	Gradient based
NSGANet [19]	3.3	88.772	94.566	87.425	96.140	EA
AmoebaNet [27]	3.2	85.949	93.498	89.222	95.953	EA
SI-EvoNet [46]	0.51	86.637	91.128	89.103	91.021	EA
AEMONet	**0.98**	**89.608**	**94.957**	90.419	**96.750**	EA

†PA represents anteroposterior view; LAT represents Lateral view
Bold highlights the performance effects of our algorithm and the corresponding improvement ratio

are resized to a uniform size of 224 × 224, and whitened and pixel normalized.

For the classification of lumbar spine X-ray images for osteoporosis diagnosis, the experimental settings of AEMONAS are kept the same as those of the above CIFAR datasets, including population size, optimizer parameters, and so on. The only difference is the hyperparameter of the architecture due to the small number of images in the lumbar spine dataset. In the search stage, the number of nodes in the two cells of the network architecture is set to 6 and 4, respectively (i.e., $k_1 = 6$ and $k_2 = 4$), and at the end of the search stage, the optimal network architecture **AEMONet** will be obtained (i.e., the K is set as 1). Meanwhile, in the retraining stage, the number of normal cells S is set to 3, the number of filters of the initial cell is set to 16, and the channel number is doubled as the spatial resolution of the reduction cell is decreased.

Since the lumbar spine dataset is not publicly benchmark and the search process of most NAS algorithms are not reproducible, the comparison algorithms we select mainly include the classic manually designed network models and the best neural architectures reported by multiple NAS algorithms, including ResNet [8], Wide ResNet [43], DenseNet [11], SENet [10], NASNet-A [53], ENAS [25], DARTS [17], NSGANet [19], AmoebaNet [27], and SI-EvoNet [46]. Table 6 shows the number of architectural parameters, and the test accuracy and AUC values on the anteroposterial view and lateral view subsets of the lumbar spine dataset achieved by the proposed AEMONAS and selected peer competitors.

As presented in Table 6, the best architecture AEMONet (as illustrated in Fig. 14) searched by the proposed AEMONAS in the lumbar spine dataset achieves 89.608% and 90.419% test classification accuracy, 94.957% and 96.750% test AUC values, respectively. This is better than all the comparison algorithms including the manually designed models and the best architectures corresponding to the NAS algorithms. As for the number of parameters of the architecture, the number of our AEMONet parameters is smaller than most of the comparison algorithms. Although SI-EvoNet has a smaller number of parameters, its classification accuracy is 2.971% and 1.316% lower than our AEMONet on the two subsets, respectively. The classification accuracy of ENAS and DARTS is very close to the AEMONet, but the number of parameters is more than two times larger than our AEMONet.

As shown in Fig. 13, the performance comparison between AEMONet and all comparison algorithms is visually illustrated (LAT subset results are taken as an example). The left figure shows the corresponding relationship between the number of all model parameters and the accuracy on test set; the architecture closer to the upper left is more outstanding. The figure on the right shows the receiver-operating characteristic curve (ROC) information of all architectures, and our proposed AEMONAS has the best ROC curve.

To further understand the mechanism behind the classification decision of the neural network, we visualize the class activation map (CAM) [50], which is widely used in image classification tasks to show the discriminative regions learned by convolutional neural architecture. As shown in Fig. 15, the examples are the class activation map on the lumbar spine dataset, where the discriminative regions learned by the model are covered by warmer colors. It can be observed that AEMONets can adaptively learn salient features of different categories of images from different perspective. For example, AEMONet can learn lesions from the overall structure of lumbar spine (Fig. 15a) and local features (Fig. 15b, c).

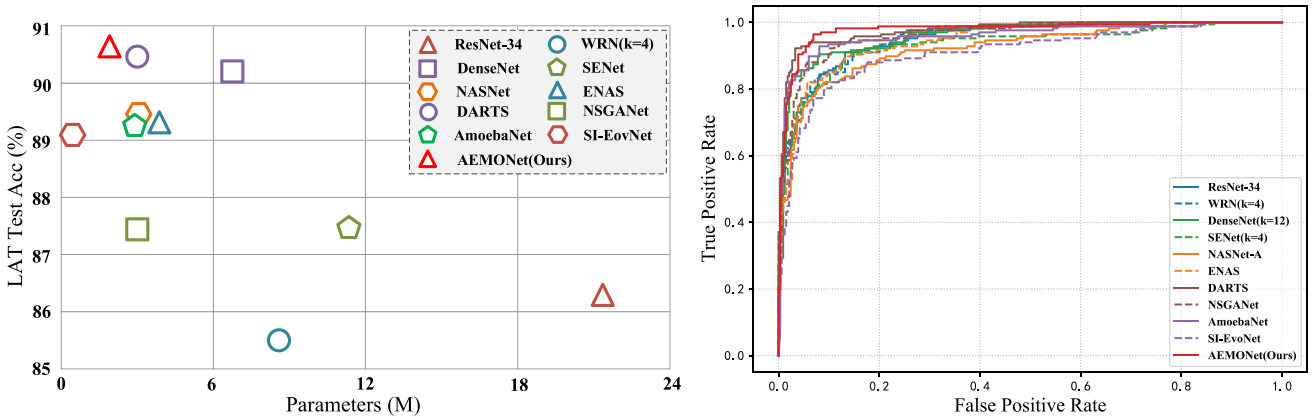
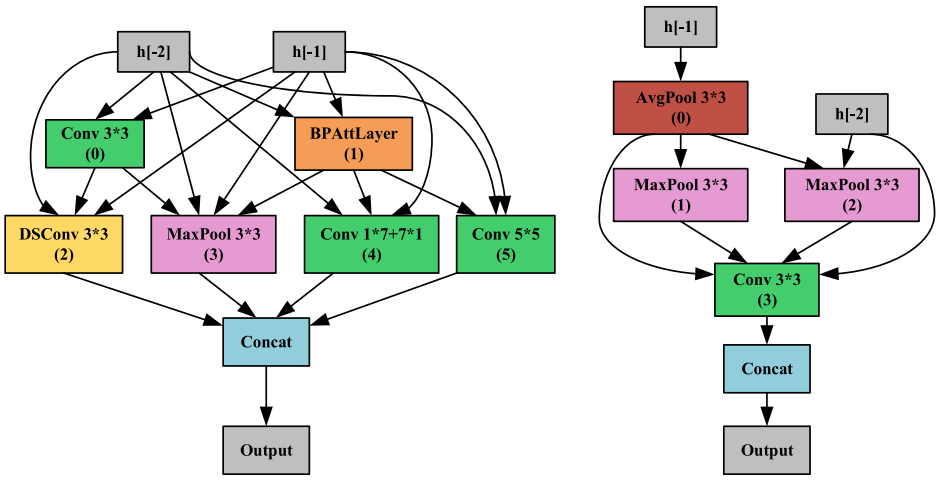


Fig. 13 left: the corresponding relationship between the number of parameters and the accuracy on LAT test set of all architectures; right: the receiver-operating characteristic curve (ROC) information of all architectures

Fig. 14 Details of normal cells and reduction cells of the AEMONet achieved by the proposed AEMONAS on lambar spine dataset



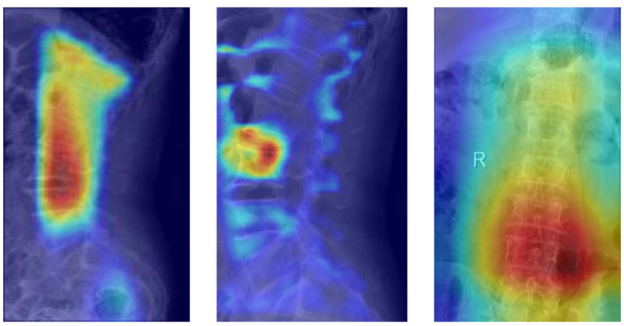


Fig. 15 Examples of the CAM of AEMONet on the lumbar spine dataset, and the red areas represent the discriminative regions learned by the neural network with BPAttLayer: **a** slight, **b** serious, and **c** slight

Conclusion

In this paper, we have proposed an effective ENAS algorithm named AEMONAS, which adopts a sampling strategy based on architecture entropy to maintain population diversity. Meanwhile, to efficiently express the search space and deeply explore the relationship between the topological structure of connections and the computing operations, the unified encoding scheme of topological structure and computing operation is suggested, and the corresponding genetic operators tailored for this encoding scheme is adopted to speedup the convergence. Experimental results on the common CIFAR-10 and CIFAR-100 datasets demonstrate the superiority of our proposed AEMONAS, and the effectiveness of the suggested strategies. Furthermore, on the practical classification problem in osteoporosis diagnosis, our proposed AEMONAS also has a good performance.

Although the proposed AEMONAS can avoid premature convergence and automatically search task-related convolutional neural architectures with excellent performance, search costs could be further reduced under tighter constraints on computing resources. Thus, we intend to improve the evaluation strategy in the future to get higher search efficiency while maintaining exploration ability. Furthermore, search space containing more elements can be designed, such as channel number, image resolution, etc.

References

1. Awad N, Mallik N, Hutter F (2020) Differential evolution for neural architecture search. arXiv:2012.06400
2. Baker B, Gupta O, Naik N, et al (2016) Designing neural network architectures using reinforcement learning. arXiv:1611.02167
3. Chen Y, Meng G, Zhang Q et al (2019) Renas: Reinforced evolutionary neural architecture search. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp 4787–4796
4. Črepinšek M, Liu SH, Mernik M (2013) Exploration and exploitation in evolutionary algorithms: a survey. ACM Comput Surv (CSUR) 45(3):1–33
5. Deb K, Pratap A, Agarwal S et al (2002) A fast and elitist multiobjective genetic algorithm: Nsga-ii. IEEE Trans Evtolution Comput 6(2):182–197
6. Elsken T, Metzen JH, Hutter F (2018) Efficient multi-objective neural architecture search via lamarckian evolution. arXiv:1804.09081
7. Gao Z, Xie J, Wang Q, et al (2019) Global second-order pooling convolutional networks. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pp 3024–3033
8. He K, Zhang X, Ren S, et al (2016) Deep residual learning for image recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 770–778
9. Hoa K, Gilberta A, Jinb H et al (2021) Neural architecture search for deep image prior. Comput Graph
10. Hu J, Shen L, Sun G (2018) Squeeze-and-excitation networks. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 7132–7141
11. Huang G, Liu Z, Van Der Maaten L et al (2017) Densely connected convolutional networks. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 4700–4708
12. Krizhevsky A, Hinton G et al (2009) Learning multiple layers of features from tiny images
13. Lee S, Choe EK, Kang HY et al (2020) The exploration of feature extraction and machine learning for predicting bone density from simple spine X-ray images in a korean population. Skeletal Radiol 49(4):613–618
14. Lin TY, RoyChowdhury A, Maji S (2015) Bilinear cnn models for fine-grained visual recognition. In: Proceedings of the IEEE international conference on computer vision, pp 1449–1457
15. Liu C, Zoph B, Neumann M, et al (2018) Progressive neural architecture search. In: Proceedings of the European conference on computer vision (ECCV), pp 19–34
16. Liu H, Simonyan K, Vinyals O, et al (2017) Hierarchical representations for efficient architecture search. arXiv:1711.00436
17. Liu H, Simonyan K, Yang Y (2018) Darts: differentiable architecture search. arXiv:1806.09055
18. Long J, Shelhamer E, Darrell T (2015) Fully convolutional networks for semantic segmentation. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 3431–3440
19. Lu Z, Whalen I, Boddeti V et al (2019) Nsga-net: neural architecture search using multi-objective genetic algorithm. In: Proceedings of the genetic and evolutionary computation conference, pp 419–427
20. Lu Z, Whalen I, Dhebar Y et al (2020) Multiobjective evolutionary design of deep convolutional neural networks for image classification. IEEE Trans Evol Comput 25(2):277–291
21. Luo R, Tian F, Qin T, et al (2018) Neural architecture optimization. arXiv:1808.07233
22. Mambrini A, Sudholt D, Yao X (2012) Homogeneous and heterogeneous island models for the set cover problem. In: International conference on parallel problem solving from nature, Springer, pp 11–20
23. Neumann F, Oliveto PS, Rudolph G, et al (2011) On the effectiveness of crossover for migration in parallel evolutionary algorithms. In: Proceedings of the 13th annual conference on Genetic and evolutionary computation, pp 1587–1594
24. Niepert M, Ahmed M, Kutzkov K (2016) Learning convolutional neural networks for graphs. In: International conference on machine learning, PMLR, pp 2014–2023
25. Pham H, Guan M, Zoph B et al (2018) Efficient neural architecture search via parameters sharing. In: International conference on machine learning, PMLR, pp 4095–4104
26. Real E, Moore S, Selle A, et al (2017) Large-scale evolution of image classifiers. In: International conference on machine learning, PMLR, pp 2902–2911
27. Real E, Aggarwal A, Huang Y, et al (2019) Regularized evolution for image classifier architecture search. In: Proceedings of the aaai conference on artificial intelligence, pp 4780–4789
28. Ren S, He K, Girshick R et al (2015) Faster r-cnn: towards real-time object detection with region proposal networks. Adv Neural Inf Process Syst 28:91–99
29. Sandler M, Howard A, Zhu M, et al (2018) Mobilenetv2: Inverted residuals and linear bottlenecks. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 4510–4520
30. Sudholt D (2020) The benefits of population diversity in evolutionary algorithms: a survey of rigorous runtime analyses. Theory Evolut Comput pp 359–404
31. Sun Y, Xue B, Zhang M et al (2019) Completely automated cnn architecture design based on blocks. IEEE transactions on neural networks and learning systems 31(4):1242–1254
32. Sun Y, Xue B, Zhang M et al (2019) Evolving deep convolutional neural networks for image classification. IEEE Trans Evol Comput 24(2):394–407
33. Sun Y, Xue B, Zhang M et al (2020) Automatically designing cnn architectures using the genetic algorithm for image classification. IEEE Trans Cybern 50(9):3840–3854
34. Sutton RS, Barto AG (2018) Reinforcement learning: an introduction. MIT Press
35. Szegedy C, Liu W, Jia Y, et al (2015) Going deeper with convolutions. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 1–9
36. Tian Y, Peng S, Yang S et al (2021) Action command encoding for surrogate assisted neural architecture search. IEEE transactions on cognitive and developmental systems
37. Tuljapurkar SD (1982) Why use population entropy? it determines the rate of convergence. J Math Biol 13(3):325–337
38. Watson RA, Jansen T (2007) A building-block royal road where crossover is provably essential. In: Proceedings of the 9th annual conference on genetic and evolutionary computation, pp 1452–1459
39. Wei J, Zhu G, Fan Z, et al (2021) Genetic u-net: automatically designed deep networks for retinal vessel segmentation using a genetic algorithm. IEEE Trans Med Imaging

40. Xie L, Yuille A (2017) Genetic cnn. In: Proceedings of the IEEE international conference on computer vision, pp 1379–1388
41. Yang S, Tian Y, Xiang X, et al (2021) Accelerating evolutionary neural architecture search via multi-fidelity evaluation. arXiv:2108.04541
42. Yang Z, Wang Y, Chen X, et al (2020) Cars: Continuous evolution for efficient neural architecture search. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pp 1829–1838
43. Zagoruyko S, Komodakis N (2016) Wide residual networks. arXiv:1605.07146
44. Zaremba W, Sutskever I, Vinyals O (2014) Recurrent neural network regularization. arXiv:1409.2329
45. Zhang B, Yu K, Ning Z et al (2020) Deep learning of lumbar spine x-ray for osteopenia and osteoporosis screening: a multicenter retrospective cohort study. Bone 140(115):561
46. Zhang H, Jin Y, Cheng R et al (2020) Efficient evolutionary search of attention convolutional networks via sampled training and node inheritance. IEEE Trans Evol Comput 25(2):371–385
47. Zhang X, Zhou X, Lin M, et al (2018) Shufflenet: An extremely efficient convolutional neural network for mobile devices. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 6848–6856
48. Zhang Y, Dai G, Zuo M et al (2019) A population entropy based adaptation strategy for differential evolution. In: Proceedings of the genetic and evolutionary computation conference companion, pp 330–331
49. Zhong Z, Yan J, Wu W, et al (2018) Practical block-wise neural network architecture generation. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 2423–2432
50. Zhou B, Khosla A, Lapedriza A, et al (2016) Learning deep features for discriminative localization. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 2921–2929
51. Zhou D, Zhou X, Zhang W, et al (2020) Econas: finding proxies for economical neural architecture search. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pp 11,396–11,404
52. Zoph B, Le QV (2016) Neural architecture search with reinforcement learning. arXiv:1611.01578
53. Zoph B, Vasudevan V, Shlens J, et al (2018) Learning transferable architectures for scalable image recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 8697–8710