International Journal of Machine Learning and Cybernetics (2022) 13:3001–3018
https://doi.org/10.1007/s13042-022-01578-8

ORIGINAL ARTICLE

Evolutionary neural networks for deep learning: a review

Yongjie Ma[1] · Yirong Xie[1]

Received: 3 December 2020 / Accepted: 9 May 2022 / Published online: 10 June 2022

Abstract
Evolutionary neural networks (ENNs) are an adaptive approach that combines the adaptive mechanism of Evolutionary algorithms (EAs) with the learning mechanism of Artificial Neural Network (ANNs). In view of the difficulties in design and development of DNNs, ENNs can optimize and supplement deep learning algorithm, and the more powerful neural network systems are hopefully built. Many valuable conclusions and results have been obtained in this field, especially in the construction of automated deep learning systems. This study conducted a systematic review of the literature on ENNs by using the PRISMA protocol. In literature analysis, the basic principles and development background of ENNs are firstly introduced. Secondly, the main research techniques are introduced in terms of connection weights, architecture design and learning rules, and the existing research results are summarized and the advantages and disadvantages of different research methods are analyzed. Then, the key technologies and related research progress of ENNs are summarized. Finally, the applications of ENNs are summarized and the direction of future work is proposed.

Keywords Deep neural networks · Evolutionary algorithms · Evolutionary neural networks · Deep learning · PRISMA review

1 Introduction

Deep learning is a core of machine learning in recent years, the development of its model has gone through two stages: shallow model and deep neural network. Since Hinton's paper [1] published in Science in 2006 solved the problem of multi-layer neural network training, the academia and industry have been enthusiastic about the research of Deep Neural Networks (DNNs) and made breakthrough progress. With the significant improvement of chip processing power and the greatly reduction of computing hardware cost, DNNs have achieved remarkable results in artificial intelligence domains such as computer vision [2, 3], natural language processing [4], transfer learning [5], malware detection [6–8], etc. However, optimal architecture design and hyper-parameters estimation of DNNs are widely considered to be intractable [9]. DNNs generally have large structures and high computational complexity. Their design is usually a manual process based on experience and expertise in the problem domains, which is tedious and time-consuming [10]. The design of DNNs also becomes more difficult with the increase of the complexity of the problem domains. The structure optimization and automatic design of DNNs have become an opportunity and challenge for artificial intelligence and attracted extensive attention from the machine learning community.

The development process of DNNs for problem domains can be divided into two parts: developing the network architecture, training and verifying it. The development of the network architecture refers to determine the topology and hyper-parameters of the neural network, and its training involves how to calculate and update the connection weights of the architecture [11]. At present, the development of DNNs is mainly based on expert, which requires designer to consider many factors (including the connection between the layers, depth, convolution calculation, etc.) and constantly tune the network according to the model performance. Therefore, DNNs often contain much redundancy and the performance has not reach their potential. The dominant methods for training neural networks are gradient-based algorithms [12], especially backpropagation, which

✉ Yongjie Ma
myjmyj@nwnu.edu.cn

Yirong Xie
yirongxie@126.com

1 College of Physics and Electronic Engineering, Northwest Normal University, Lanzhou 730070, China

have been performing well in machine learning. However, gradient-based algorithms proceed with directional moves, which makes them very sensitive to the initial solution and local optimum. On the other hand, the fitness functions for real-world applications trend to be discontinuous or multi-modal, and the optimization problem can be straightforward, difficult, or misleading [13].

Some typical methods to optimize DNNs have been proposed, including manual design methods such as Network Pruning and automatic design methods based on Neural Architecture Search. Evolutionary neural networks (ENNs), also known as NeuroEvolution (NE), are an automated method that use Evolutionary algorithms (EAs) to optimize DNNs. EAs [14] are a traditional optimization algorithm that are gradient-free and insensitive to the local optimal. They search without directed moves of the gradients-based algorithms, and have great advantages in dealing with complex and non-convex problems [9]. Aiming at the difficulties in the design and development of DNNs, EAs can optimize and supplement them from three aspects: network training, architecture design and learning rules. The origin of ENNs can be traced back to the 1980s, when they were used in the study of shallow neural networks. Advances in computing power have enabled researchers to revitalize the old algorithms, and the dramatic increase in the complexity of DNNs has put new technical requirements on ENNs, spurring a great deal of research into the evolution of DNNs. The greater charm of ENNs lies in the combination of EAs and artificial neural networks (ANNs), which make the development process of the system similar to the generation and learning of biological brain. In this way, ENNs are closer to the natural brain-like evolution, and can fundamentally change the design of the whole automatic machine learning system.

ENNs for deep learning have attracted more and more attention. Stanley et al. [15] summarized the contribution of ENNs to meta-learning and architecture search in the review article, and proposed that ENNs are an important tool in the long-term pursuit of artificial general intelligence. Many valuable conclusions and results have been obtained in this field, especially in the construction of automated deep learning systems. Performance comparable to state-of-the-art work has been obtained in architectural search for convolutional neural networks for image classification. The purpose of this paper is to summarize the progress of these ENNs' work in a more comprehensive way, to summarize the work on the dramatic increase of architecture optimization, and to highlight the research and continuous exploration of the weights and learning rules of neural networks by EAs to promote the development of fusion techniques for different optimization contents. In addition, to provide reference and inspiration for interested researchers, this paper discusses the key technologies of ENNs research and summarizes the corresponding research progress, and puts forward the future research direction.

This review adopted the PRISMA (Preferred Reporting Items for Systematic Reviews and Meta-Analyses) protocol (see http://prisma-statement.org), specified the research objectives, keywords, and inclusion criteria for the research questions. The PRISMA selection process of relevant literature is shown in Fig. 1.

In identification stage, the following keywords were used to develop an initial search criterion: "Evolutionary Neural Network", "Neuroevolution", "Evolutionary architecture search". The keyword search was conducted on the university library databases in Nov. 20, 2020, and obtained 1330 results. The number of articles shows a dramatic increase in number from 2017 to 2020 when we did this survey, with the main focus reflecting papers on evolutionary neural architecture search. In addition, we included articles published up to Jan. 2022 in the revision of this paper. The search engine of databases covered over 400 different bibliographic repositories, including Scopus, ScienceDirect, Web of Science, Directory of Open Access journals, et al.

In screening stage, after excluding duplicate articles, the number of articles was 1181. Then, the articles were screened for the research field, we selected articles from fields related to computer science and removed articles from other fields such as biology, mechanical engineering, physics, basic medicine, etc. The number of articles was reduced to 710.

Next, the articles were screened for eligibility, the 384 articles of full-text available online journal and conference were selected. Journals and conferences where the key authors cited and published on this topic are mainly identified, and articles were screened accordingly.

In included stage, the key authors related to evolutionary neural networks were first identified, and the articles were

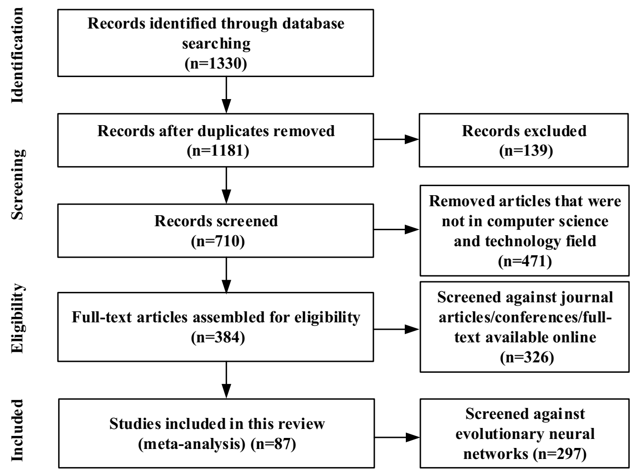
Identification
Records identified through database searching (n=1330)
Screening
Records after duplicates removed (n=1181)
Records excluded (n=139)
Records screened (n=710)
Removed articles that were not in computer science and technology field (n=471)
Eligibility
Full-text articles assembled for eligibility (n=384)
Screened against journal articles/conferences/full-text available online (n=326)
Included
Studies included in this review (meta-analysis) (n=87)
Screened against evolutionary neural networks (n=297)

Fig. 1 The PRISMA selection process of relevant literature

classified to identify the research branches. The categories are then filtered and verified, classified and finalized. The articles were selected based on their titles, abstracts and keywords (Optimization of deep neural network structures based on evolutionary algorithms). The articles that match the theme of this review were included, and the number of articles was finally reduced to 87.

87 articles were analyzed using descriptive methods of explanation construction and pattern matching. The selected articles were analyzed according to three categories with different optimization objectives of network weights, architectures and learning rules to assess the similarities and differences between algorithms. The reviewed articles were categorized into specific subject categories, and the specific optimization strategies and progress under different categories were further analyzed and summarized.

The remainder of this paper is organized as follows. Section 2 briefly reviews the Evolutionary neural networks, including Evolutionary algorithms, Artificial neural networks, Evolutionary neural networks, and their training process. Section 3 introduces the theory and research of Evolutionary neural networks from the three main optimization aspects of neural networks. Section 4 discusses the key technologies of Evolutionary neural networks. Section 5 summarizes the paper with an outlook for future research.

2 The basic principles

ENNs are composed of ANNs and EAs. In order to facilitate the comprehension of the reader, this section first gives a basic description of the basic principles and developed process of ANNs. Second, to illustrate the necessity of structural optimization of DNNs, this section provides statistics on the development background and research limitations of DNNs. Then, this section gives a basic introduction to EAs as an optimization algorithm. Finally, this section details how EAs are utilized to optimize DNNs.

2.1 The basic principles of ANNs

ANNs are inspired by the process of natural neuron processing information, and a general neuron-like processing unit is defined as shown in Fig. 2. The computational between the inputs and output can be written out as Eq. (1).

$$y_j = f\left(b_j + \sum_{i=1}^{n} (x_i * w_{ij})\right) \tag{1}$$

where y_j is the output to the unit, x_i is the input, w_{ij} is the weights of x_i and unit j, b_j is the bias, $f(\cdot)$ is the activation function. A neural network is a combination of lots of these neuron-like processing units. A simple ANN consists of

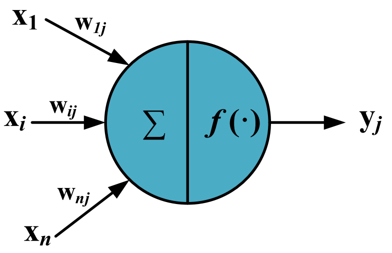

Fig. 2 Neuron-like processing unit

input layer, hidden layers, and output layer [16]. In the learning process, the input layer units are used to take the values of the input features, and the input values are processed by the hidden layer and the output layer units, then the result is output by the output layer units.

The development process of a DNNs algorithm requires designing the network architecture according to the target environment, including the number of network layers, the number of neurons, the connection method, the activation function, et al. The training of the network is also a key step that will update the weights for the neurons and effect the performance of ANNs. Gradient-based algorithms are currently the most widely used training algorithms, which are used to adjust the connection weights to minimize the difference between the actual and desired output vectors of the network [17].

2.2 The development background of DNNs

The shallow ANNs with simple structure cannot meet the increased sample size and characteristic dimension in practical engineering applications. In 2006, Professor Hinton [1] from the University of Toronto in Canada proposed a solution to the problem of gradient disappearance in deep network training: they combined with unsupervised pre-training to initialize weights and supervised training fine-tuning, which solved the problem of parameters training of DNNs. After that, various DNN models have sprung up, stimulated the enthusiastic for the development of deep learning in the academic world and industry. AlexNet [18] won the first prize in the ImageNet Large Scale Visual Recognition Competition (ILSVRC) in 2012, and the performance of the CNNs continued to rise in the following years. ResNet-50 [22] achieved a recognition accuracy rate of 96.4\% in ILSVRC 2015. The artificial intelligence Go program AlphaGo [23], developed by Google DeepMind based on deep reinforcement learning, achieved a 99.8\% winning rate against other Go programs and defeated the human European Go champion 5–0.

The deep learning methods [24] can be divided into supervised learning, semi-supervised learning, and unsupervised learning, where supervision refers to whether the data set used has detailed annotation information. Supervised learning uses the labeled training datasets, mainly including Convolutional Neural Network (CNN) and Recurrent Neural Network (RNN), etc. Semi-supervised learning uses partial-labeled datasets, and the methods include Deep Reinforcement Learning (DRL) and Generative Adversarial Networks (GAN). Unsupervised learning usually includes Clustering, Dimensionality Reduction and Generative Models. In the field of deep learning, CNN and RNN are most widely used.

An important feature of network performance improvement is the increase of model complexity and training parameters. The CNN generally increases depth by increasing the number of convolutional layers, and the deep models obtained in this way have better performance in classification tasks [25]. As shown in Table 1, the performance of CNN keeps increasing, while its network complexity and parameters are also rise correspondingly. Specifically, as the recognition error rate in the ImageNet dataset keeps decreasing, the number of network layers in CNN keeps increasing, and the number of training parameters also keep high. Therefore, DNN architecture engineering is more and more huge, which stimulates the development of automating architecture engineering. The complexity of problem domain and the wide application of DNN are increasing, but the experience of most end users is limited. Moreover, based on the requirement of real-world application, it is a difficulty to find the most accurate and effective architectures in a reasonable time based on the manual design.

2.3 Evolutionary algorithms

EAs are a class of stochastic search and optimization procedures inspired by the natural evolution mechanisms of biological systems [26]. The general framework of ENNs is shown in Algorithm 1. During the evolution process [27], the population is initialized according to the encoding strategy (line 1), where the parameters (connection weights, network architecture or learning rules) to be optimized are encoded in individuals. Then the evolution operations are performed (lines 3–9), then individuals are evaluated and assigned a fitness according to their ability to solve the optimization problems [27]. The individuals with high fitness for genetic operations (selection, crossover, mutation, etc.) to form the next generation. This process is an intelligent parallel search for better genotypes and continues until predefined termination conditions are met and the best individual is finally produced (line 10) [28, 29]. Through the evolutionary process, beneficial traits are passed down through iteration to promote the population's adaptation to the environment, thus generating the optimal solution to the problem. EAs are independent of gradient information and can solve a wide range of complex, nonlinear, non-differentiable and multimodal optimization problems [30].

```
Algorithm 1: The framework of ENNs
1   $P_0$ ←Initialize the population with the encoding strategy;
2    t ← 0;
3   while termination criterion is not satisfied do
4       Evaluate the fitness of individuals in $P_t$;
5       S ← Select parent solutions from Pt according to the fitness;
6       $Q_t$ ← Generate offspring from S by the genetic operators;
7       $P_{t+1}$ ←Create the next population by selection from $P_t \cup Q_t$;
8       t ← t + 1;
9   End
10  Output the best individual from $P_t$
```

EAs include Genetic Algorithm (GA), Evolution Strategy (ES), Genetic Programming (GP), and Evolutionary Programming (EP), etc. GA uses genetic operators such as selection, crossover, and mutation to search for the optimal genetic vector from the search space [31]. ES only uses mutation operators to evolve the real vector solution, and the tree-based programming structure is applied to express the genetic information [32]. GP searches for the optimal program structure from the topological search space of a computer program [33], EP is used for the evolution of computer program parameters that the structure is unchanged [34]. Although these algorithms focus on different concepts, but they are essentially based on the idea of evolution.

2.4 Evolutionary neural networks

ENNs combines the adaptive mechanism of EAs with the learning mechanism of ANNs, and has the adaptability to

Table 1 Classic convolutional neural networks and corresponding parameters

Method	Year	Network layers	Convolution layers	Parameters		Top-5 Error rate (%)
				Convolution layer (M)	Fully connected layer (M)	
AlexNet [18]	2012	8	5	2.3	58.6	16.4
Overfeat [19]	2013	8	5	16	130	14.2
VGGNet-16 [20]	2014	16	13	14.7	124	7.4
GoogLeNet [21]	2014	22	21	6	1 M	6.7
ResNet-50 [22]	2015	50	49	23.5	2	3.6

dynamic environment. ENNs algorithm consists of three main components: search space, evolutionary computation, and performance evaluation strategy. Search space is designed by defining the optimization objective (weights, architecture or learning rules of the networks) and encoding method, then the evolutionary operators are designed to guide the evolutionary search and the adaptive optimization process is executed. At the same time, the performance of the evolved network in the target environment is used as the fitness evaluation to guide the evolution process, so that the performance of the evolved network in the terminal environment can be steadily improving.

Given the search space, the neural network structure can be encoded as an expression under that space. The size of the search space determines the difficulty of the network structure search. Various constraints can be introduced according to the prior knowledge of specific tasks to restrict the basic units to form local search space, in which the search cost and complexity can be significantly reduced, but the diversity of network structures will be reduced. The search space contains the initial population of EAs, and generally represents the most primitive solutions. The candidate networks of various sizes can be created to improve the diversity of the population [35]. Various approaches to enhance the initial population have been utilized in literature [36–38]. Moreover, the training cost of DNN is a difficult problem in the fitness evaluation of ENNs, the complete training of all individuals in the population requires huge computational resources. As the key technologies of ENN, various fitness evaluation and accelerated evaluation methods are discussed in Sects. 4.2 and 4.4 respectively.

Figure 3 represents the general process of the ENNs algorithm. First, the initialized population is generated according to the encoding method in search space, then the codes of the individuals in the population are decoded to the corresponding DNNs, the performance evaluation strategy is set and the DNNs are evaluated according to the terminal environment, the individuals are selected according to the fitness values, the population operations are performed on them, new individuals are generated. With the iteration of this process, the genetic operator searches for new network structures, the selection operator guides the evolutionary direction of the population, and the individuals in the population are continuously optimized until the maximum number of evolutionary generations is satisfied. Finally, the optimal individual is selected and decoded into the corresponding network structure, and the optimization process is completed.

Evolutionary operations include crossover and mutation: In mutation, a single individual is input, its gene is modified to produce new traits. Mutation operators include the addition, deletion, or replacement of gene locus, which constitute the local search of individuals. A Gaussian mutation [39] can be used to guide the mutation, which can predict which architecture may be good, and the new generated individuals are sampled in the regions of the search space where the fitness values are likely to be high [40]. In crossover, two or more individuals are input as the parent genes to recombine and produce a new offspring, which constitute the global search of the algorithm. There are various crossover methods based on different evolutionary algorithms, but the general rule is to select gene loci and exchange part of gene coding of individuals. Sun et al. [29] used the simulated binary crossover to combine coding parameters from two matching layers. Sapra et al. [41] proposed a disruptive crossover that swapped the whole cluster between the two individuals at corresponding positions. There is a problem of competing conventions in the crossover operation. Since the same solution can be represented in more than one way, crossing different genes of the same solution is unlikely to produce offspring with the same or better performance. Measures such as historical markings need to be taken to keep the crossover operation in order.

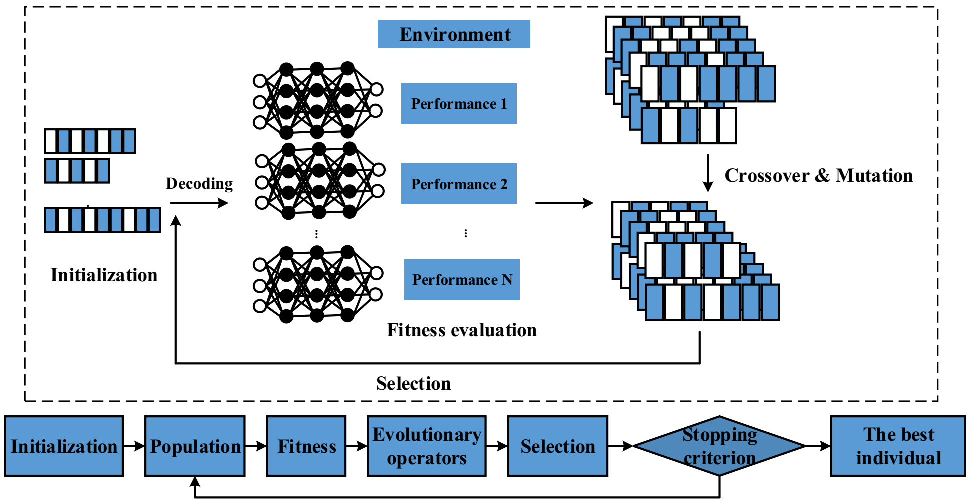

Fig. 3 The general process of the ENNs algorithm

3 Evolution of neural networks

According to the development of ANNs, EAs can optimize them in three parts: connection weights, network structure and learning rules, which are important research components of ENNs. ENNs have obtained some advanced research results in these aspects, especially in the architecture optimization component. In this section, we present the theory of ENNs and the technical progress of them in the field of deep learning from each of these three aspects. In particular, the components of ENNs are not limited to independent optimization. Connection weights can be combined with structural evolution to automatically design the trained network, while learning rules can be integrated into structural evolution to achieve plasticity in ANNs. Therefore, some research works optimize multiple components simultaneously, and these works are also presented in this paper.

3.1 Evolution of connection weights

The evolution of weights was initially used for ANNs training and searching optimal connection weights for ANN with a predetermined topology. This process is usually expressed as minimize network errors, such as the mean square error between the actual and expected output of the networks. The dominated training algorithm is are based on gradient minimization, but they are easy to be trapped in local optimum. One way to overcome the drawbacks of gradient-based algorithms is to train with EAs, which are a black-box methods that are flexible and gradient-free. In the evolution of connection weights, the environment is defined by the predetermined architecture and the specific task. The weights can be represented as a string of binary values or a string of real numbers [42]. This method has higher efficiency and scalability, and can produce better results than BP algorithm. Weights evolution initially focused on only the weights of small, fixed-topology networks [43].

In modern deep learning algorithms, EAs also perform very well in training neural networks for DRL. Open AI Labs [44] used ES to train DNNs, disturbed the current set of parameters and checked their performance to make them move towards higher reward direction, which directly searched the parameter space of the DNNs. This method is invariant to action frequency and delayed rewards, and it does not need temporal discounting or value function approximation [45]. Uber AI Labs [46] evolved the weights of a DNN with the gradient-free GA, the experiment results showed that GA performed well on hard DRL problems, particularly in the challenging field of Atari. The evolved competitive Atari-playing agents enabled a 10,000-fold compression of DNN. The unexpected competitive performance of the GA suggests that following the gradient is not the best choice for optimizing performance in some cases, because the distribution of solutions of increasing quality is unexpectedly dense. The deep GA encoding method was described as Eq. (2) in [46].

$$\theta^n = \psi\left(\theta^{n-1}, \tau_n\right) = \theta^{n-1} + \sigma\varepsilon\left(\tau_n\right) \quad (2)$$

where θ is the parameter vector of each individual, θ^n is an offspring of θ^{n-1}, ψ (θ^{n-1},τ_n) is a deterministic mutation function, τ is the encoding of θ^n consisting of a list of mutation seeds, $\theta^0=\phi(\tau_0)$, where ϕ is a deterministic initialization function, and $\varepsilon(\tau_n) \sim N(0, 1)$ is a deterministic Gaussian pseudo-random number generator with an input seed τ_n that produces a vector of length $|\theta|$.

The efficiency of training can be improved by incorporating EAs with Gradient descent algorithms. Gradient descent algorithms are the dominant algorithms for training DNNs, they have good local search performance [47]. EAs have a good performance of global searching, but their search ability near the global optimal solution is weak. The complementarities stimulate the research of hybrid algorithm of the two algorithms, which will improve the overall performance of the algorithm by taking advantage of their respective advantages. One issue that needs to be weighed carefully in designing this hybrid algorithm is how to design the switch between the two algorithms. Cui et al. [48] optimized in a framework by alternating SGD and evolutionary steps to improve the average fitness of the population, which ensured that the best fitness in the population will never degrade. This hybrid algorithm is also commonly used in reinforcement learning domain to generate higher sample efficiency and faster learning [49, 50], in which a reinforcement agent is trained by using diverse data provided by the population and inserted into the EA population regularly to infuse the gradient information. Yang et al. [51] designed genetic operators and used gradient information to directly optimize the weights of DNNs. Also, the sparsity of the network was optimized along with the training loss, which reduced the complexity of the network and alleviated the overfitting.

3.2 Evolution of network architectures

The evolution of network architectures is to design the optimal architectures for the target domain, including topologies and hyper-parameters. The network architecture has a crucial effect on its performance, the simple network has strong limitations in dealing with problems, while the overly complex network will reduce the generalization [9]. The performance of many advanced networks depends mainly on structural innovations. The general, architecture optimization problem can be formulated as Eq. (3).

$$\arg\max ACC\left(A_{\lambda,\omega}\right) \quad s.t.\lambda \in \Lambda \tag{3}$$

Where $A_{\lambda,\ \omega}$ is the architecture with the parameters λ and weights ω; λ refers to the architecture definition, such as the number and configurations of layers; ω is updated by training the model on the datasets. Λ is the architecture parameter space, which is the set of all possible parameter configurations under the encoding mode; $ACC(\cdot)$ is the accuracy on the validation set of A_λ.

The encoding of DNN can be divided into direct encoding and indirect encoding. All connections and nodes of the networks are encoded to genes in direct encoding, including graph encoding and binary encoding, where genotype corresponds to phenotype one to one. For large-scale DNNs, directly encoded gene sequences will generate huge search space, resulting in low efficiency of structural search [52]. In indirect encoding, only some features or generation rules of network structures are encoded into genes, the structure expressed is more compact than direct coding. The evolution of network architectures consists of three objectives: (1) optimizing the configuration of DNNs; (2) automatically design the DNNs; (3) DNNs compression.

1. Optimizing the configurations of DNNs

The configuration of DNNs architectures include topology and hyper-parameters. There are still technical deficiencies in the existing technologies, the redundant parameters in the topology will increase the over-fitting and wasted of computing resources, and many weak neurons can be excluded without damaging the overall performance [53]. To optimize the network structure, it is necessary to consider the accuracy of the network and the sparsity of the connection. Liu et al. [54] proposed a hierarchical structure learning method, based on multi-objective optimization, to find the optimal structure that is as sparse as possible and maintains sufficient performance. Kim et al. [55] proposed the neural evolution multi-objective optimization (NEMO) algorithm, and realized the automatic optimization of CNNs accuracy and speed by using MOEAs. Zhou et al. [56] proposed a surrogate-assisted evolutionary search method for optimization of the hyperparameters and neural architecture of the reservoir of liquid state machines using the covariance matrix adaptation evolution strategy (CMA-ES).

In supervised learning, the hyper-parameters must be configured prior to running for best predictive performance [57]. The basic methods of hyper-parameter optimization are grid search [58] and random search [59]. The methods are simple and easy to use, but its computational cost will be staggering when the parameter dimensions grows and the single computation is high. In order to deal with the high dimensional parameter space and the large model that is difficult to evaluate quickly, more methods are needed to do the hyper-parameter search. CMA-ES [60] is an evolutionary algorithm successfully applied to optimize the hyper-parameters of DNNs, which is competitive especially in the regime of parallel evaluations. In literature [61], an adaptive activation function is constructed by ENNs, while the selected basic functions by GA and the learned combination coefficients are adapted to the input data, the network optimizer and learning rate are also optimized.

2. Automatically design network architectures

(a) Automatic design of DNNs. The automatic design of network architectures is the dominant goal for structure evolution, the topologies and parameters are optimized simultaneously to produce the trained networks directly. As early as 2002, Stanley and Miikkulainen [62] proposed the NeuroEvolution of Augmenting Topologies (NEAT) algorithm to evolve the architectures and connection weights of small-scale neural networks. NEAT is a powerful method for artificially evolving neural networks, and demonstrates that evolving topology along with weights can be made a major advantage. However, the neural network at that time was shallow. The development of modern computing resources stimulated the development of large-scale learning systems. ENNs can also take advantage of modern computing resources, in combination with the long-term technological advances made in EAs, to design large-scale network capable of handling large amounts of data and challenging tasks [63].

Miikkulainen et al. [64] proposed CoDeepNEAT, an automatic method to optimize deep learning architecture through evolution, in which NEAT algorithm was first extended to DeepNEAT that evolved the topology and hyper-parameters of the deep neural network. The genes locus of DeepNEAT no longer represented neurons, but a layer in DNNs, which contained real values and hyper-parameters tables. These hyper-parameters determined the layer type and

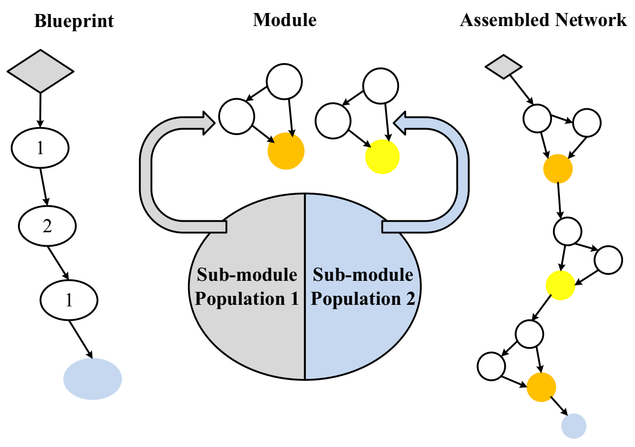

Fig. 4 An assembled network of CoDeepNEAT

layer properties such as the number of neurons and activation function of the layer. DeepNEAT was then extended further to the co-evolution of modules and blueprints, as shown in Fig. 4, modules and blueprints were assembled into a network through replacement of blueprint nodes with corresponding modules. This method has been verified in standard benchmark tasks of target recognition, language modeling and in practical applications of subtitled images on magazine websites. In literature [65], a deep evolutionary network structure representation (DENSER) is proposed that can automatically design DNN based on two different levels of representation. The outer layer encodes the general structure of the network, and the inner layer encodes the parameters related to each layer. The speed of the model is greatly increased by enabling the layers of the evolved networks to be connected to any of the previous layers, it is easy to promote the emergence of skip connections by this way [66].

(b) Constrained search space for CNN. The computer vision domain has been one of the most important research topics in deep learning [67], where most widely used computer vision algorithm is the CNN, especially in the large-scale image classification problems. Based on the expert networks, some block-based ENNs algorithms [68–70] have been designed to automatically design CNN and improve the efficiency of traversing the search space. A series of predefined blocks composed of convolution or pooling operations are optimized as the minimum unit in these methods. As illustrated in Fig. 5, on the left are two different design blocks. Each node in the graphs corresponds to a layer in a neural network, e.g., a convolutional or pooling layer. Different layer types are visualized by different colors. On the right is an architecture built by stacking the blocks sequentially. Note that blocks can also be combined in a more complex manner, such as increasing branches and skip connections, and the layers can be simply replaced by blocks.

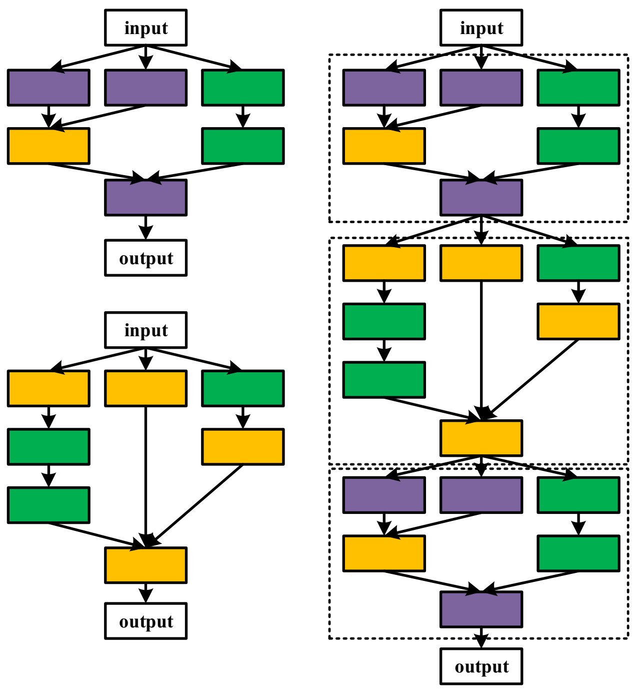

Fig. 5 An illustration of the block-based search space

Suganuma et al. [71] used Cartesian genetic programming (CGP) to automatically construct CNN architecture for image classification task, which adopted highly functional modules as the node functions in CGP, such as convolutional blocks and tensor concatenation. Sun et al. [72] evolved CNN architectures based on ResNet and DenseNet blocks, the proposed algorithm is completely automatic in designing CNN architectures and consumes much less computational resource to find the best CNN architectures. The block-based methods can greatly reduce the search space of network structure, but also has limitations for the exploration of non-convolution modules of network structure. The Large-Scale Evolution of Image Classifiers [73] employed simple evolutionary techniques at unprecedented scales to discover models for the CIFAR-10 and CIFAR-100 datasets, this approach used as little prior heuristics as possible to evolve a network structure from scratch and achieved competitive performance. Zhang et al. [74] proposed an evolutionary strategy that can be executed with limited computing resources to find better topologies for deep CNNs, which is also essentially free in evolving network structures.

(c) Search for novel architectures. Evolutionary optimization of network structure promises to produce optimal architectures different from human design. Desell [75] presented Exploration of Augmenting Convolutional Topologies (EXACT) algorithm that firstly used for the evolution of arbitrarily structured CNN. This algorithm developed topology based on free structure, the showed organic structures are significantly different from standard human designed architectures. EXACT is also an alternative method for evolutionary recurrent neural networks (RNN) through evolutionary exploration of augmenting Long Short Term Memory (LSTM) Topologies [76]. The Evolutionary Exploration of Augmenting Memory Models (EXAMM) [77] algorithm was presented under the framework of EXACT, which evolved RNN using a wide variety of memory structures, such as RNN, GRU, LSTM, MGU and UGRNN. The algorithm could yield performant architectures but the reliability was poor in the average and worst cases.

Real et al. [78] presented new image classification architectures through regularized evolution, which achieved the optimal architecture comparable to human design and reached the state of the art for ImageNet, the experiment showed that regularized evolution consistently produced models with similar or higher accuracy, across a variety of contexts without need for re-tuning parameters. Aimed at the optimization of the basic structure of LSTM nodes, Rawal and Miikkulainen [79] explored new variations through the evolution of a tree-based encoding of the gated memory nodes. This method discovered nodes with multiple recurrent paths and multiple memory cells, which led to significant improvement in the standard language modeling benchmark task.

(d) Comparison with other optimization strategies. In the field of architecture optimization, many recent research results have emerged in neural architecture search based on different optimization strategies, mainly including reinforcement learning (RL)-based, gradient-based algorithms and EAs-based in this paper. In order to compare the performance of these algorithms, this paper summarizes some recent methods that have performed experiments on the popular datasets CIFAR10 and CIFAR100 for image classification tasks and compares the advantages and disadvantages in terms of architectural parameters, performance and time consumption of the algorithms, as shown in Table 2. The performance metric is taken as classification accuracy. The time consumption metric taken is the GPU days [29], the unit GPU day means the algorithm has performed one day on one GPU when the algorithm terminates. The model parameters need to be taken into account when comparing model performance, as models with larger number of parameters usually achieve better performance.

The RL-based algorithms include NAS V3 [80], Efficient NAS [81], Block-QNN-S [82], MetaQNN [83], DPP-Net [84], PPP-Net-A [85], and Proxyless NAS [86]. Gradient-based methods include DARTS [87], RC-DARTS-C42 [88], RC-DARTS-C14 [88], SNAS [89]. Evolutionary neural architecture algorithms include Hierarchical Evolution [46], Large-scale Evolution [73], CGPCNN [68], AE-CNN [72], CNN-GA [90], AE-CNN + E2EPP [27], and SI-EvoNet [17].

As can be seen from Table 2, these state-of-the-art neural architecture search algorithms have competitive advantages in terms of performance, but they differ significantly in terms of algorithm time consumption. Since gradient-based algorithms can converge faster, they all consumed less processing time. In RL-based and EAs-based methods, the definition of the search space and the performance evaluation of the architectures greatly affect the time consumption of the algorithms.

In large-scale evolution, genotypes in the search space are encoded with only layers and connections. Large-scale evolution is able to search for architectures different from those designed by humans, but it requires large-scale distributed computing to achieve it. As mentioned in Sect. 3.2 (b) Constrained search space, other algorithms optimize the consumption time of the algorithm by limiting the search space according to the expert networks. This constraint enables the algorithm to optimize the search based on the expert

Table 2 Performance comparison of different neural architecture search algorithms on CIFAR datasets

Category	Architecture	CIFAR10	CIFAR100	parameters	GPU days
RL-based	NAS V3	95.53	–	7.1 M	22,400
RL-based	Efficient NAS	97.06	–	4.2 M	0.5
RL-based	Block-QNN-S	96.70	82.95	6.1 M	90
RL-based	MetaQNN	93.08	72.86	11.2 M	90
RL-based	DPP-Net	94.16	–	0.45 M	8
RL-based	PPP-Net-A	94.72	–	0.45 M	–
RL-based	Proxyless NAS	97.92	–	5.7 M	1500
GD-based	DARTS	97.18	–	3.4 M	1
GD-based	RC-DARTS-C42	97.19	82.46	3.3 M	1
GD-based	RC-DARTS-C14	95.83	–	0.43 M	1
GD-based	SNAS	97.15	–	2.8 M	1.5
EAs-based	Hierarchical evolution	96.37	–	15.7 M	300
EAs-based	Large-scale evolution	94.60	–	5.4 M	2750
EAs-based	Large-scale evolution	–	77.00	40.4 M	2750
EAs-based	CGP-CNN	94.02	–	1.7 M	3150
EAs-based	CNN-GA	96.78	–	2.9 M	35
EAs-based	CNN-GA	–	79.47	4.1 M	40
EAs-based	AE-CNN	95.3	–	2.0 M	27
EAs-based	AE-CNN	–	77.6	5.4 M	36
EAs-based	AE-CNN + E2EPP	94.70	–	4.3 M	7
EAs-based	AE-CNN + E2EPP	–	77.98	20.9 M	10
EAs-based	SI-EvoNet	96.02	–	0.51 M	0.46
EAs-based	SI-EvoNet	–	79.16	0.99 M	0.81

network, but gives up the search for novel architectures. In addition, AE-CNN + E2EPP introduced an end-to-end performance evaluation strategy to optimize the adaptation evaluation of DNNs, which further reduced the consumption time of the algorithm. SI-EvoNet used sampling training and node inheritance adapted to search for node connections within blocks. In sampling training, individuals of each generation were trained with a small number of epochs, and the weights of the individual nodes trained during evolution are inherited directly to offspring. SI-EvoNet took up the least GPU days.

3. DNNs Compression

Another goal of DNNs optimization is to minimize the complexity or size of the network while maximizing its performance [93], breaking through its application limitations in the environment with high real-time requirements and resource constraints. Real-world applications for deep learning are becoming increasingly important and need to run on smartphones, however, the hundreds of millions of weights of modern DNNs cannot fit to the few gigabytes of RAM in most smartphones. Multi-objective evolutionary optimization algorithm [94, 95] can be used to find sufficiently small and practical configuration to achieve the balance between the size and accuracy of the network structure [96]. Loni et al. [97] built a highly optimized network using a multi-objective evolutionary approach, which considered accuracy and network size as two objectives and designed a search space inspired by a dense architecture. Cetto et al. [98] used NSGA-II algorithm to optimize and trade-off the performance of CNNs in terms of accuracy and size, which was aimed at searching for smaller, mobile size networks. This experiment uncovered that small changes in network architecture can significantly reduce its size while retaining the major part of its accuracy.

3.3 Evolution of learning rules

The evolution of learning rules is to learn how to learn, which make the evolved neural network systems adapt to the dynamic environment. A training algorithm works differently when it is applied to networks with different architectures. It is difficult to design an optimal learning rule when prior knowledge about the structure of ANNs is limited, so it is best to develop an automatic system that adapts the learning rule to the architecture and the problem to be solved [11]. In natural biology, learning (i.e., the effects of environment on phenotypes) and development (i.e., the effects of genotypes on phenotypes) are closely related. EAs is the most natural way to study the plasticity of neural networks, which can realize the integration of evolutionary and learning processes like natural organisms [99].

The evolution may be restricted to a simple parameter search, or search a larger space of arbitrary and general plasticity rules. Plasticity may also be applied to all or parts of the network, thus effectively implementing the evolution of learning architectures [100]. A learning rule can be described by Eq. (4).

$$\Delta\omega(t) = \sum_{k=1}^{n} \sum_{i_1,i_2,\ldots,i_k=1}^{n} \left(\theta_{i_1,i_2,\ldots,i_k} \prod_{j=1}^{k} x_{i_j}(t-1) \right) \tag{4}$$

where t is time. Δw is the weight change, x_1, x_2, …, x_n are local variables, and 's are real coefficients. The major aim of the evolution of learning rules is to decide these coefficients.

The early studies on the evolution of learning rules were to optimize the parameters of learning rules for fixed or hand-designed ANN architectures [101, 102]. Abraham [63] proposed Meta-Learning Evolutionary Artificial Neural Network (MLEANN) algorithm, an automatic computational framework for the adaptive optimization of ANNs, parameters (such as the neural network architecture, activation function, connection weights and learning algorithm) were self-adjusted according to the problem. Niv et al. [103] evolved synaptic plasticity dynamics of the Hebbian learning rule, gave rise to varying exploration levels and the well-documented choice strategies of risk aversion and probability matching.

The prospect for the evolution of learning rules is to design the plasticity neural systems for lifelong learning. Neuroplasticity is a way to improve learning performance, and two of the most widely studied mechanisms of neuroplasticity, synaptic and intrinsic plasticity, are currently in operation. Synaptic plasticity is the most common and important mechanism of neuroplasticity. Several computational models of neuroplasticity have been proposed, including the Hebbian rule [104], the BCM rule [105], the anti-Oja rule [106], and spike-time-dependent plasticity [107], among others. The interplay between synaptic plasticity and intrinsic plasticity contributes to the adaptation of the nervous system to internal and external environmental changes. Triesch [108] confirmed that intrinsic plasticity plays an important role in shaping the activation function of neurons to maintain the balance of the final output of neurons.

The learning principles can be discovered in the evolutionary process, so that knowledge and skills can be acquired through the interaction with the environment [109]. In this case, the connection weights of the DNNs will change over the lifetime of the network, and evolution determines not only the architecture and weights, but also the rules that guide how and when specific weights change. Networks evolved in this way are, in principle, more like biological brains, which change over the course of their lives in response to their experiences [110]. The mapping between genotype and phenotype (i.e., ontogeny) is influenced by both genotype and external environment and takes place over the lifetime of the individual [99]. Risi and Stanley

[111, 112] proposed the Hypercube-based NeuroEvolution of Augmenting Topologies (HyperNEAT) [113] method, combined plasticity with indirect encoding, which allowed not only patterns of weights across the connectivity of an ANN to be generated by a function of its geometry, but also patterns of arbitrary learning rules. Wang et al. [114] proposed a collaborative plasticity learning rule to adapt energy storage connections in echo state networks (ESNs), which considered not only the regulation of synaptic weights but also the tuning of intrinsic excitability of neurons. State-of-the-art performance was obtained using this network on a number of prediction and classification benchmark problems. The design of learning rules still relies heavily on expert design, and the use of EAs for the optimization of learning rules is still in the development stage.

4 The key techniques of evolutionary neural networks

The design of high-performance ENNs algorithms requires many factors to be considered, which are reflected in the design of the search space, evolutionary operators, network performance evaluation, etc. This section summarizes the key technologies for ENNs, and the breakthroughs in these technologies are promising for further development to ENNs. This section summarizes the recent technological advances in ENNs in terms of key technologies and aims to provide a reference for interested researchers.

4.1 Gene encoding of topology

The representation under the search space determines the number of variables and the correlation between them, so genetic encoding is critical to the topological evolution and the features that can be represented [115]. Genetic encoding involves how to describe the optimization objectives of DNNs for EAs, which is the first problem to be solved in the development of ENNs. The human brain has about 100 trillion connections and 100 billion neurons [116], and its genes are encapsulated in a DNA-based genetic code [15]. Designing an efficient encoding method that can encapsulate the modules and connection patterns of DNNs into EAs, just like the way human brain genes, which is the most breakthrough technology for ENNs to realize artificial general intelligence.

The known genetic encoding strategies are based on three types of representations: grid, geometric and indirect representations, which are shown in Fig. 6. The gird representation is based on binary encoding, which can represent the topological details without the limitation of the discrete representation, but its decision space has a stairs-like boundary and the number of its decision variables will become difficult to calculate for large-scale problems [117]. In geometric representation, the genotype encodes the geometry of the structures within the design domain, and the structures are composed of primitives that can be related as nodes and edges of a graph [118]. In this way, the individual can be encoded into a graph as DNA [65, 66] for evolve large free-structure networks, but this direct encoding still requires a lot of computing resources. Indirect representation encodes a variable generative structure, where the genome is a formula for generating a network rather than a direct description of the network itself [119, 120] Indirect encoding makes for reasonably compact genomes and enables very large neural networks to be evolved through compact encoding [15]. A modern popular indirect encoding is Compositional Pattern Producing Networks (CPPNs) [121], which compresses a pattern with regularities and symmetries into a relatively small set of genes. A successful example of CPPNs is the HyperNEAT algorithm, which can generate functional neural networks with hundreds of thousands to millions of connections, and multiple variants [122, 123] were proposed to evolve even larger scale neural networks motivated by the HyperNEAT.

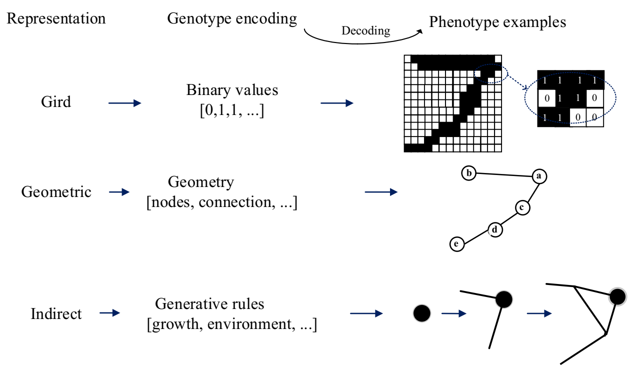
Representation
Genotype encoding
Decoding
Phenotype examples
Gird
Binary values [0,1,1, ...]
Geometric
Geometry [nodes, connection, ...]
Indirect
Generative rules [growth, environment, ...]

Fig. 6 Three types of representations in gene encodings

Evolutionary efficiency can be improved by constrained search space, such as the block-based methods. The modules with relatively high functions such as convolution block and tensor concatenation were adopted as node functions in CGP-CNN [68], this encoding method can significantly improve the performance of EAs and represent the network structure with variable length and skip connection, which is very flexible. In fact, the optimal depth of DNNs also needs to be explored during the evolution, Sun et al. [25, 65] proposed an effective variable-length genetic encoding strategy to represent different building blocks and unpredictable optimal depth in CNNs, which can explore the optimal depth of the network and save computing resources.

4.2 Fitness evaluation and novelty search

Another major factor affecting performance of ENNs is the selection of individuals for reproduction. Fitness evaluation is a traditional method for selection, which evaluates the fitness of individuals in the target environment and guides the evolutionary algorithm to converge to the optimal solution. The fitness of neural networks is generally based on the mean square error of target output and actual output, and the evaluation of network complexity is often added to optimize the structure. Due to the depth of DNNs, it will take a lot of time and computing resources to fully train and evaluate individuals, which is the main reason for the reduced efficiency. More effective model evaluation mechanism needs to be explored to improve the efficiency of ENNs. An effective method is to investigate only the tendency of the performance [29], that is, an individual is trained with a small number of epochs, and then the fitness value is calculated by combining the classification errors and the parameters of network structure. However, this method also requires a huge amount computing resources to process large-scale data. Stork et al. [124] proposed a surrogate model based on phenotypic distance measurement to partially replace fitness function, which utilized the L1 norm distance to account for large dimensions of the input and output vector and significantly improved evaluation efficiency for tasks with expensive evaluations.

Selection operations based on individual fitness mean that more suitable individuals have a higher chance of reproducing, but such selection leads to a decrease in population diversity and does not put pressure on the population to greatly improve. Novelty Search (NS) [125, 126] is a method that abandons objectives, it rewards neural networks only for behaving in a way that is novel relative to individuals generated earlier in search. NS can avoid deception and foster the emergence of adaptive behavior [127, 128]. The performance of NS in DRL is better than that of objective-oriented search. Uber AI labs [129] combined NS and ES for Atari and simulated robot learning to bypass the deception trap, the experiment results proved that the new algorithm synthesized could avoid the local optimal encountered by ES and achieve higher performance.

4.3 Modular neural network

The structure and performance of DNNs are closely correlated, the co-evolution of modules in DNNs has been proved to be a successful method to improve the performance of ENNs [130–132]. Modular neural network is a neural network consisting of multiple sub-modules, which can decompose complex problems into finite sub-problems, so that each sub-problem can be processed by sub-modules, and then an integration mechanism is used to integrate sub-modules to solve complex problems. By sharing modules and co-evolution, modularity can accelerate evolution and facilitate the research of multitasking learning, thus facilitating the development of DNNs and designing complex systems. Praczyk [133] proposed the cooperative co-evolutionary neural networks, in which each module network evolved in a different population and formed a complete modular artificial neural network. The method is effective but sensitive to the size of modules and the generated architecture is too complex, resulting in poor generalization effect. Liang et al. [134] used the custom routing evolution method of shared modules to multitask learning, and evolved different topologies for different tasks and tied these tasks together by sharing modules among them. This method has strong cooperative ability and greatly improves the research level of multitask learning.

Another exciting prospect of modular neural network is to relieve the catastrophic forgetting, i.e., agents learn a new skill do not lose previously acquired skills. Ellefsen et al. [135] combined neuromodulation with modular neural network, selectively changed learning rate of each neural connection according to the environment, thus reduced the interference between the learning task. The experiments proved that modularity can improve performance by retaining skills. However, the connection cost technique was separately encoded for each connection, which made the control of connections in evolutionary processes a challenging optimization problem. Velez et al. [136] proposed diffusion-based neuromodulation, this work simulated the release of chemicals that modulate learning of spatial region in neural networks, which enabled evolution to generate task-specific local learning and functional modularization. In addition to modularity, more objectives related to network structure need to be widely explored. Literature [137] defined objectives as alignment with user-recommended decomposition patterns and high diversity of populations in decomposition patterns, which performed well on problem with a very clear and decomposable structure.

4.4 Speeding up computation

One of the main challenges of ENNs for deep learning is the considerable computation by searching large spaces of complex models and evaluating networks. Evolution is a parallel search method, and when it is combined with the acceleration provided by modern computers and matched with large datasets, the performance of the algorithm can be greatly improved. Liang et al. [10] distributed the evaluation of individuals in each generation of the population on hundreds of GPU devices through cloud computing, and this parallel evaluation enabled thousands of candidate networks to be trained in a few days, thus made architecture search easy to process. Real et al. [73] deployed evolution of network structure on 250 computers to perform optimization of image classification in parallel. EXACT algorithm [75] trained more than 120,000 CNN with more than 4500 volunteer computers, and these large-scale calculations achieved considerable success.

Even with the aid of a GPU, ENNs algorithms often takes a lot of time to find a valid model, and more work is focused on the evolution of DNNs with little computing power, even on a single GPU. The acceleration of evolution can be designed in the whole process, including preprocessing data, encoding gene, designing evolutionary operators, networks training and evaluating, etc. One way to preprocess data is to first create a convolutional auto-encoder to effectively compress the input data [138], which saves computing resources in network training and helps in limiting networks size. The combination of traditional optimization methods can also provide acceleration for architecture search. Suganuma et al. [58] used a model training dataset to train the network with the usual stochastic gradient descent method, and then took the performance on the architecture evaluation dataset as the fitness of the architecture. Liu et al. [139] explored promising architectures through performance migration, which trained on smaller datasets and transferred the explored network to the large datasets.

The speeding up strategies in different processes of ENNs from are reviewed in Table 3, some of which have been mentioned in the corresponding sections. Sun et al. [59] introduced a local search strategy and variable-length encoding, which enabled to evolve a DNN with a lot of parameters under limited computing resources. Assunção et al. [140] used mutation operator to modify the maximum training time of individuals, so that the evaluation time increased with the complexity of the network, which could effectively generate fully trained DNNs at the end of evolution. Li et al. [141] proposed an automatic method for designing CNNs architectures that can search for optimal network models meeting the preset constraint, in which an adaptive penalty algorithm was used for fitness evaluation, and a selective repair operation was developed for infeasible individuals to search for feasible CNN architectures.

The optimization on performance evaluation is a trendy research direction in current accelerated computing, and mainly includes performance prediction and weight inheritance. Performance predictors replace the time-consuming training process by predicting the fitness values of DNNs. Performance prediction models can eliminate the need to train candidate architectures and enable transfer learning between datasets [142].

The end-to-end performance predictor collects a set of DNNs with corresponding performance, and then constructs a model to map the DNN architecture and performance values. When the evolutionary process generates a new DNN, the regression model will directly predict its performance. Deng et al. [143] encoded the individual layers into vectors and predicted individual accuracy based on LSTM. Istrate et al. [144] predicted the network accuracy by analyzing the network structure and the training dataset characterization number (peak accuracy obtained by training a deeply normalized ProbeNet on the dataset for 10 epochs). Sun et al. [91] validated the effectiveness and efficiency of using random forest based performance predictors in ENAS and proposed a novel training protocol [145], the ranking information between two training samples was used to train the regression model. Learning curve-based performance predictors predict the performance by fitting the learning curve of each DNN [146–148], and smoothed performance curves are required to build proper regression models in such methods.

In weight inheritance strategies, the networks are avoided to be trained from scratch and computational resources are saved. Zhang et al. [92] improved the evolutionary efficiency in the evolutionary neural architecture search, the offspring individuals inherited the trained node information from their parents. In addition, Zhang et al. [149] proposed a partial weight sharing one-shot NAS framework. During the evolutionary search, the crossover operator randomly sampled nodes from the parent individuals to construct new candidate architectures. The mutation operator enhanced evolutionary exploration by replacing selected nodes of a one-time model with randomly generated nodes. The algorithm achieved performance comparable to state-of-the-art architectures in image classification tasks. Elsken et al. [150] proposed a Lamarckian inheritance mechanism for multi-objective architecture search, in which generated children networks were started with the predictive performance of their trained parents.

Table 3 The speeding up strategies in different processes of ENNs

Process	Speeding up approaches
Search space	Restricting the search space based on prior knowledge The variable-length genetic encoding strategy
Evolutionary computing	Optimized evolutionary operators
Performance Evaluation	Performance prediction Weight inheritance

4.5 Parameter value

The parameters of the EAs (population size, mutation rate, crossover rate, maximum generation, etc.) also need a proper choice to ensure its accuracy and efficiency, but it often needs to undergo time consuming parameter adaptation in practice. An approach for parameter optimization [151] is bi-level optimization [152, 153] for parameter self-adaptation, in which the lower-level objective function is the performance of the parameters discovered by the optimization algorithm, and the upper-level objective function is the performance of algorithm given parameters, this method can effectively find the optimal parameters of ENNs algorithm.

5 Summary and prospect

This study has conducted a systematic review of the literature on ENNs by using the PRISMA protocol, and a detailed analysis is conducted from three components according to the development of DNNs. Key technologies and relevant literatures are discussed and analyzed. More powerful deep learning architectures can be built. ENNs have shown good performance in many tasks such as character recognition, speech recognition, image classification and deep reinforcement learning [154], especially in neural architecture search for image classification tasks, and promoted the development of automatic deep learning. ENNs have also been applied in many practical fields, including robot design [155, 156], data prediction [157–159], medical system [160, 161] and transportation system [162–164], etc. In the long term, the evolved automated systems have the potential to replace artificial design algorithms to design artificial intelligence products.

One of the challenges and opportunities for ENNs is to build an open process that triggers the evolution of an increasingly complex set of brain-like nervous systems. The research status of ENNs in algorithm improvement and application shows that it has become one of the hot spots in the field of computational intelligence. ENNs have made a lot of research results and have been widely used, their strong ability since their emergence is enough to make us believe that they will have a larger research space and a wider application field. Further work in the future is particularly worth exploring in the following areas.

(a) One of the advantages of ENNs is that they can use the long-standing success and experience of EAs and ANNs. Especially, ENNs algorithm of deep learning requires huge computational resources, the combination with other automatic search architecture methods is one of the possible important exploration directions. In addition, the concept of ENNs is not limited to the optimization of neural networks by EAs, but also extends to other population-based heuristic algorithms, such as particle swarm algorithm [165], ant colony algorithm [166], and artificial bee colony algorithm [167], etc.

(b) The generalization ability of the system can be improved by structural goals such as modularization. Multi-objective based training allows a neural network to evolve to deal with two or more targets related to the DNN, such as approximate error, network complexity, input dimension, etc. In addition, optimization decision based on multiple candidate objects can be added [31].

(c) The plasticity of evolutionary learning rules is still concentrated in relatively small ANNs. The development of larger plasticity systems enables DNNs to learn throughout their lives, which can fundamentally change the automatic design of the entire learning system in deep learning. Therefore, a better understand the dynamics of the interaction between evolution and learning is needed to continue to explore evolutionary representations of learning mechanisms.

(d) Due to the hardware limitations caused by the limited memory of mobile and embedded devices, the compromise between the network size and accuracy of the architecture is also the key to the wide application of the system. The improvement of algorithms and hardware will make the program run better. Simple and accurate goals can be used to guide evolution, including training time, execution time or network memory, calculation cost, etc.

Acknowledgements This work is supported by the NSFC (National Natural Science Foundation of China) project (Grant number: 62066041, 41861047) and the Northwest Normal University young teachers' scientific research capability upgrading program (NWNU-LKQN-17-6).

References

1. Hinton GE, Salakhutdinov RR (2006) Reducing the dimensionality of data with neural networks. Science 313(5786):504–507
2. Yang F, Zhang L, Yu S et al (2020) Feature pyramid and hierarchical boosting network for pavement crack detection. IEEE Trans Intell Transp Syst 21(4):1525–1535
3. Guo Y, Liu Y, Oerlemans A et al (2016) Deep learning for visual understanding: a review. Neurocomputing 187:27–48
4. Ahmad F, Abbasi A, Li J et al (2020) A deep learning architecture for psychometric natural language processing. ACM Trans Inf Syst (TOIS) 38(1):1–29
5. Knoll F, Hammernik K, Yi Z (2019) Assessment of the generalization of learned image reconstruction and the potential for transfer learning. Magn Reson Med 81(1):116–128
6. Mahindru A, Sangal AL (2021) MLDroid—framework for Android malware detection using machine learning techniques. Neural Comput Appl 33:5183–5240
7. Mahindru A, Sangal AL (2021) DeepDroid: feature selection approach to detect android malware using deep learning. In: Proceedings of the 2019 IEEE 10th International Conference on software engineering and service science (ICSESS). https://doi.org/10.1109/ICSESS47205.2019.9040821
8. Mahindru A, Sangal AL (2020) PerbDroid: effective malware detection model developed using machine learning classification techniques. J Towards Bio-inspir Tech Softw Eng. https://doi.org/10.1007/978-3-030-40928-9_7

9. Sun Y, Yen GG et al (2019) Evolving unsupervised deep neural networks for learning meaningful representations. IEEE Trans Evol Comput 23(1):89–103
10. Liang J, Meyerson E, Hodjat B, Fink D, Mutch K, Miikkulainen R (2019) Evolutionary neural AutoML for deep learning. In: Proceedings of the genetic and evolutionary computation conference (GECCO), pp 401–409
11. Fernandez-Blanco E, Rivero D, Gestal M et al (2015) A Hybrid evolutionary system for automated artificial neural networks generation and simplification in biomedical applications. Curr Bioinform 10(5):672–691
12. Lehman J, Chen J, Clune J, Stanley KO (2018) Safe mutations for deep and recurrent neural networks through output gradients. In: Proceedings of the genetic and evolutionary computation conference (GECCO), pp 117–124
13. Lu G, Li J, Yao X (2014) Fitness landscapes and problem difficulty in evolutionary algorithms: from theory to applications. In: Richter H (ed) Recent advances in the theory and application of fitness landscapes. Springer Berlin Heidelberg, Berlin, pp 133–152
14. Whitley D (2001) An overview of evolutionary algorithms: practical issues and common pitfalls. Inf Softw Technol 43(14):817–831
15. Stanley KO, Clune J, Lehman J et al (2019) Designing neural networks through neuroevolution. Nat Mach Intell 1(1):24–35
16. Alvaro M, Joaquin D, Ronald J et al (2002) Systematic learning of gene functional classes from DNA array expression data by using multilayer perceptrons. Genome Res 12(11):1703–1715
17. Rumelhart DE, Hinton GE, Williams RJ (1986) Learning representations by back-propagating errors. Nature 323(6088):533–536
18. Krizhevsky A, Sutskever I, Hinton GE (2017) ImageNet classification with deep convolutional neural networks. Commun ACM 60(6):84–90
19. Zeiler MD, Fergus R (2013) Visualizing and understanding convolutional networks. arXiv preprint, arXiv:1311.2901
20. Simonyan K, Zisserman A (2014) very deep convolutional networks for large-scale image recognition. arXiv preprint, arXiv 1409.1556
21. Szegedy C, Liu W, Jia Y, Sermanet P, Reed S, Anguelov D, Erhan D, Vanhoucke V, Rabinovich A (2015) Going deeper with convolutions. In: Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)
22. He K, Zhang X, Ren S, Sun J (2016) Deep residual learning for image recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)
23. Silver D, Huang A, Maddison CJ et al (2016) Mastering the game of Go with deep neural networks and tree search. Nature 529(7587):484–489
24. He KM, Sun J (2015) Convolutional neural networks at constrained time cost. In: Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)
25. Lecun Y, Bengio Y, Hinton G (2015) Deep learning. Nature 521(7553):436
26. Ding S, Li H, Su C et al (2013) Evolutionary artificial neural networks: a review. Artif Intell Rev 39(3):251–260
27. Jin Y, Jürgen B (2005) Evolutionary optimization in uncertain environments-a survey. IEEE Trans Evol Comput 9(3):303–317
28. Sharkey N (2002) Evolutionary computation: the fossil record. IEE Rev 45(1):40–40
29. Sun Y, Xue B, Zhang M (2017) Evolving deep convolutional neural networks for image classification. IEEE Trans Evol Comput 24(2):394–407
30. Bonissone PP, Subbu R, Eklund N et al (2006) Evolutionary algorithms + domain knowledge = real-world evolutionary computation. IEEE Trans Evol Comput 10(3):256–280
31. Ojha VK, Abraham A, Snasel V (2017) Metaheuristic design of feedforward neural networks: a review of two decades of research. Eng Appl Artif Intell 60:97–116
32. Grefenstette JJ (1988) Genetic algorithms and machine learning. Mach Learn 3(2):95–99
33. Hansen N (2006) The CMA evolution strategy: a comparing review. Stud Fuzziness Soft Comput 192:75–102
34. Banzhaf W, Koza JR (2000) Genetic programming. IEEE Intell Syst 15(3):74–84
35. Lehman J, Miikkulainen R (2013) Neuroevolution. Scholarpedia 8(6):30977
36. Dufourq E, Bassett B (2017) Automated problem identification: regression vs classification via evolutionary deep networks. In: Proceedings of the South African institute of computer scientists and information technologists
37. Sharma D, Deb K et al (2011) Domain-specific initial population strategy for compliant mechanisms using customized genetic algorithm. Struct Multidiscip Optim 43(4):541–554
38. Madeira JA, Rodrigues HC et al (2006) Multiobjective topology optimization of structures using genetic algorithms with chromosome repairing. Struct Multidiscip Optim 32(1):31–39
39. Lorenzo PR, Nalepa J (2018) Memetic evolution of deep neural networks. In: Proceedings of the genetic and evolutionary computation conference (GECCO)
40. Liu Y, Sun Y, Xue B, Zhang M, Yen GG, Tan KC (2021) A survey on evolutionary neural architecture search. IEEE Trans Neural Netw Learn Syst. https://doi.org/10.1109/TNNLS.2021.3100554
41. Sapra D, Pimentel AD (2020) An evolutionary optimization algorithm for gradually saturating objective functions. In: Proceedings of GECCO
42. Montana D, Davis L et al (1989) Training feedforward neural networks using genetic algorithms. In: Proceedings of the international joint conference on Artificial intelligence (IJCAI). vol 1, pp 762–767
43. Yao X (1999) Evolving artificial neural networks. In: Proceedings of the IEEE, 87(9):1423–1447
44. Lehman J, Chen J, Clune J et al (2018) ES is more than just a traditional finite-difference approximator. In: Proceedings of the genetic and evolutionary computation conference (GECCO)
45. Salimans T, Ho J, Chen X, et al. (2017) evolution strategies as a scalable alternative to reinforcement learning. arXiv preprint, arXiv:1703.03864
46. Such F P, Madhavan V, Conti E, et al. (2017) Deep neuroevolution: genetic algorithms are a competitive alternative for training deep neural networks for reinforcement learning. arXiv preprint, arXiv: 1712.06567
47. Singh GAP, Gupta PK (2018) Performance analysis of various machine learning-based approaches for detection and classification of lung cancer in humans. Neural Comput Appl 31:6863–6877
48. Cui X, Zhang W, Tüske Z, Picheny M (2018) Evolutionary stochastic gradient descent for optimization of deep neural networks. In: Proceedings of the 32nd international conference on neural information processing systems (NIPS)
49. Khadka S, Tumer K (2018) Evolution-guided policy gradient in reinforcement learning. In: In: Proceedings of the 32nd international conference on neural information processing systems (NIPS)
50. Houthooft R, Chen Y, Isola P, Stadie B, Wolski F, Jonathan H,OpenAI, Abbeel P (2018) Evolved policy gradients. In:

Proceedings of the 32nd international conference on neural information processing systems (NIPS)
51. Yang S, Tian Y, He C, Zhang X, Tan KC, Jin Y (2021) A Gradient-guided evolutionary approach to training deep neural networks. IEEE Trans Neural Netw Learn Syst. https://doi.org/10.1109/TNNLS.2021.3061630
52. Liu H, Simonyan K, Vinyals O, et al (2018) Genetic programming approach to designing convolutional Architecture Search. In: Proceedings of the ICLR
53. Hu H, Peng R, Tai YW, et al (2016) Network trimming: a data-driven neuron pruning approach towards efficient deep architectures. arXiv preprint, arXiv:1607.03250
54. Liu J, Gong M, Miao Q et al (2018) Structure learning for deep neural networks based on multiobjective optimization. IEEE Trans Neural Netw Learn Syst 29(99):2450–2463
55. Kim YH, Reddy B, Yun S, Seo C (2017) Nemo: neuro-evolution with multiobjective optimization of deep neural network for speed and accuracy. In: Proceedings of the international conference on machine learning (ICML)
56. Zhou Y, Jin Y, Ding J (2020) Surrogate-assisted evolutionary search of spiking neural architectures in liquid state machines. Neurocomputing 406:12–23
57. Probst P, Bischl B, Boulesteix AL (2018) Tunability: Importance of hyperparameters of machine learning algorithms. arXiv preprint, arXiv:1802.09596
58. Ghawi R, Pfeffer J (2019) Efficient hyperparameter tuning with grid search for text categorization using kNN approach with BM25 similarity. Open Comput Sci 9(1):160–180
59. Bergstra J, Bengio Y (2012) Random search for hyper-parameter optimization. J Mach Learn Res 13(1):281–305
60. Loshchilov I, Hutter F (2016) CMA-ES for hyperparameter optimization of deep neural networks. arXiv preprint, arXiv:1604.07269
61. ZahediNasaba R, Mohsenia H (2020) Neuroevolutionary based convolutional neural network with adaptive activation functions. Neurocomputing 381:306–313
62. Stanley KO, Miikkulainen R (2002) Evolving neural networks through augmenting topologies. Evol Comput 10(2):99–127
63. Abraham A (2004) Meta learning evolutionary artificial neural networks. Neurocomputing 56:1–38
64. Miikkulainen R, Liang J, Meyerson E, et al (2017) Evolving deep neural networks. arXiv preprint, arXiv:1703.00548
65. Assunção F, Lourenço N, Machado P et al (2018) DENSER: deep evolutionary network structured representation. Genet Program Evol Mach 20:5–35
66. Assuno F, Loureno N, Machado P et al (2019) Fast DENSER: efficient deep neuroevolution. Genetic Programming 11451:197–212
67. Minar M R, Naher J (2018) Recent advances in deep learning: an overview. arXiv preprint, arXiv:1807.08169
68. Suganuma M, Shirakawa S, Nagao T (2017) A genetic programming approach to designing convolutional neural network architectures. In: Proceedings of the genetic and evolutionary computation conference, pp 497–504
69. Ma B, Li X, Xia Y et al (2020) Autonomous deep learning: a genetic DCNN designer for image classification. Neurocomputing 379:152–161
70. Xie L, Yuille A (2017) Genetic CNN. In: Proceedings of the IEEE international conference on computer vision (ICCV), pp 1379–1388
71. Suganuma M, Kobayashi M, Shirakawa S et al (2020) Evolution of deep convolutional neural networks using Cartesian genetic programming. Evol Comput 28(1):141–163
72. Sun Y, Xue B, Zhang M et al (2019) Completely automated CNN architecture design based on blocks. IEEE Trans Neural Netw Learn Syst 31(4):1–13
73. Real E, Moore S, Selle A, Saxena S, Suematsu YL, Tan J, Le QV, Kurakin A (2017) Large-scale evolution of image classifiers. In: Proceedings of the international conference on machine learning (ICML)
74. Zhang H, Kiranyaz S, Gabbouj M (2018) Finding better topologies for deep convolutional neural networks by evolution. arXiv preprint, arXiv:1809.03242
75. Desell T (2017) Large scale evolution of convolutional neural networks using volunteer computing. In: Proceedings of the the genetic and evolutionary computation conference companion (GECCO)
76. ElSaid A, Wild B, Higgins J, Desell T (2016) Using LSTM recurrent neural networks to predict excess vibration events in aircraft engines. In: Proceedings of 2016 IEEE 12th international conference on e-Science (e-Science), pp 260–269
77. Ororbia A, ElSaid A E, Desell T (2019) Investigating recurrent neural network memory structures using neuro-evolution. In: Proceedings of the genetic and evolutionary computation conference (GECCO)
78. Real E, Aggarwal A, Huang Y, Le QV (2018) Regularized evolution for image classifier architecture search. In: Proceedings of the AAAI conference on artificial intelligence. vol 33(01), pp 4780–4789
79. Rawal A, Miikkulainen R (2018) From nodes to networks: evolving recurrent neural networks. arXiv preprint, arXiv:1803.04439
80. Zoph B, Le Q V (2016) Neural architecture search with reinforcement learning. arXiv preprint, arXiv:1611.01578
81. Pham H, Guan M, Zoph B, Le Q, Dean J (2018) Efficient neural architecture search via parameters sharing. In: Proceedings of the 35th International conference on machine learning, PMLR 80, pp 4095–4104
82. Zhong Z, Yang Z, Deng B, Yan J, Wu W, Shao J, Liu C (2021) Blockqnn: efficient block-wise neural network architecture generation. IEEE Trans Pattern Anal Mach Intell 43(7):2314–2328
83. Baker B, Gupta O, Naik N, Raskar R (2016) Designing neural network architectures using reinforcement learning. arXiv preprint arXiv:1611.02167
84. Dong J, Cheng A, Juan D, Wei W, Sun M (2018) Dpp-net: device-aware progressive search for pareto-optimal neural architectures. In: Proceedings of the 6th international conference on learning representations (ICLR). https://openreview.net/forum?id=B1NT3TAIM
85. Dong J, Cheng A, Juan D, Wei W, Sun M (2018) Ppp-net: platform-aware progressive search for pareto-optimal neural architectures. In: Proceedings of 6th international conference on learning representations (ICLR)
86. Cai H, Zhu L, Han S (2018) Proxylessnas: direct neural architecture search on target task and hardware. arXiv preprint arXiv:1812.00332
87. Liu H, Simonyan K, Yang Y (2018) Darts: differentiable architecture search. arXiv preprint arXiv:1806.09055
88. Jin X, Wang J, Slocum J, Yang M, Dai S, Yan S, Feng J (2019) Rc-darts: resource constrained differentiable architecture search. arXiv preprint arXiv:1912.12814
89. Xie S, Zheng H, Liu C, Lin L (2018) Snas: stochastic neural architecture search. arXiv preprint arXiv:1812.09926
90. Sun Y, Xue B, Zhang M, Yen GG, Lv J (2020) Automatically designing CNN architectures using the genetic algorithm for image classification. IEEE Trans Cybern 50(9):3840–3854
91. Sun Y, Wang H, Xue B, Jin Y, Yen GG, Zhang M (2019) Surrogate-assisted evolutionary deep learning using an end-to-end

random forest-based performance predictor. IEEE Trans Evol Comput 24(2):350–364
92. Zhang H, Jin Y, Cheng R, Hao K (2020) Efficient evolutionary search of attention convolutional networks via sampled training and node inheritance. IEEE Trans Evol Comput 25(2):371–385
93. Howard AG, Zhu M, Chen B, KalenichenkoD, Wang W, Weyand T, Andreetto M, Adam H (2017) MobileNets: efficient convolutional neural networks for mobile vision applications. arXiv preprint. arXiv:1704.04861
94. Ming F, Gong W, Wang L (2022) A two-stage evolutionary algorithm with balanced convergence and diversity for many-objective optimization. IEEE Trans Syst Man Cybern Syst. https://doi.org/10.1109/TSMC.2022.3143657
95. Sun Y, Xue B, Zhang M, Yen GG (2019) A new two-stage evolutionary algorithm for many-objective optimization. IEEE Trans Evol Comput 23(5):748–761
96. Huang J, Sun W, Huang L (2020) Deep neural networks compression learning based on multi-objective evolutionary algorithms. Neurocomputing 378:260–269
97. Loni M, Sinaei S, Zoljodi A, Daneshtalab M, Sjödin M (2020) DeepMaker: a multi-objective optimization framework for deep neural networks in embedded systems. Microprocess Microsyst 73:102989
98. Cetto T, Byrne J, Xu X et al (2019) Size/accuracy trade-off in convolutional neural networks: an evolutionary approach. In: Proceedings of the INNSBDDL
99. Nolfi S, Miglino O, Parisi D (1994) Phenotypic plasticity in evolving neural networks. In: Proceedings of the PerAc'94. From perception to action, pp 146–157
100. Soltoggio A, Stanley KO, Risi S (2018) Born to learn: the inspiration, progress, and future of evolved plastic artificial neural networks. Neural Netw 108:48–67
101. Chalmers DJ (1991) The evolution of learning: An experiment in genetic connectionism. In: Connectionist models. Morgan Kaufmann, Elsevier, pp 81–90
102. Kim HB, Jung SH, Kim TG et al (1996) Fast learning method for back-propagation neural network by evolutionary adaptation of learning rates. Neurocomputing 11(1):101–106
103. Niv Y, Joel D, Meilijson I et al (2002) Evolution of reinforcement learning in uncertain environments: a simple explanation for complex foraging behaviors. Adapt Behav 10(1):5–24
104. Hebb DO (1949) The organization of behavior: a neuropsychological theory. Wiley, New York
105. Bienenstock EL, Cooper LN, Munro PW (1982) Theory for the development of neuron selectivity: orientation specificity and binocular interaction in visual cortex. J Neurosci 2(1):32–48
106. Babinec Š, Pospíchal J (2007) Improving the prediction accuracy of echo state neural networks by anti-Oja's learning. In: Proceedings of the International Conference on artificial neural networks, Springer, pp 19–28
107. Bi GQ, Poo MM (1998) Synaptic modifications in cultured hippocampal neurons: dependence on spike timing, synaptic strength, and postsynaptic cell type. J Neurosci 18(24):10464–10472
108. Triesch J (2005) A gradient rule for the plasticity of a neuron's intrinsic excitability. In: Proceedings of the artificial neural networks: biological inspirations(ICANN). vol 3696, Springer, pp 65–70
109. Coleman OJ, Blair AD (2012) Evolving plastic neural networks for online learning: review and future directions. In: Proceedings of the Australasian joint conference on artificial intelligence, pp 326–337
110. Stanley KO (2017) Neuroevolution: a different kind of deep learning. Obtenido de. 27(04):2019
111. Risi S, Stanley KO (2014) Guided self-organization in indirectly encoded and evolving topographic maps. In: Proceedings of the 2014 annual conference on genetic and evolutionary computation. ACM, pp 713–720
112. Risi S, Stanley KO (2010) Indirectly encoding neural plasticity as a pattern of local rules. In: Proceedings of the international conference on simulation of adaptive behavior, pp 533–543
113. Stanley KO, Ambrosio DBD, Gauci J (2009) A hypercube-based encoding for evolving large-scale neural networks. Artif Life 15(2):185–212
114. Wang X, Jin Y, Hao K (2021) Synergies between synaptic and intrinsic plasticity in echo state networks. Neurocomputing 432:32–43
115. Guirguis D et al (2020) Evolutionary black-box topology optimization: challenges and promises. IEEE Trans Evol Comput 24(4):613–633
116. Azevedo FAC, Carvalho LRB, Grinberg LT et al (2010) Equal numbers of neuronal and nonneuronal cells make the human brain an isometrically scaled-up primate brain. J Comp Neurol 513(5):532–541
117. Boichot R et al (2016) A genetic algorithm for topology optimization of area-to-point heat conduction problem. Int J Therm Sci 108:209–217
118. Aulig N, Olhofer M (2016) Evolutionary computation for topology optimization of mechanical structures: An overview of representation. In: Proceedings of the 2016 IEEE congress on evolutionary computation (CEC), pp 1948–1955
119. Gruau F (1993) Genetic synthesis of modular neural networks. In: Proceedings of the GECCO
120. Gruau F, Whitley D, Pyeatt L (1996) A comparison between cellular encoding and direct encoding for genetic neural networks. In: Proceedings of the 1st annual conference on genetic programming, pp 81–89
121. Stanley KO (2007) Compositional pattern producing networks: a novel abstraction of development. Genet Program Evolvable Mach 8(2):131–162
122. Pugh JK, Stanley KO (2013) Evolving multimodal controllers with hyperneat. In: Proceedings of the 15th annual conference on Genetic and evolutionary computation, pp 735–742
123. Fernando C, Banarse D, Reynolds M et al (2016) Convolution by evolution: differentiable pattern producing networks. In: Proceedings of the Genetic and Evolutionary Computation Conference (GECCO), 109-116
124. Stork J, Zaefferer M, Bartz-Beielstein T (2019) Improving neuroevolution efficiency by surrogate model-based optimization with phenotypic distance kernels. In: Proceedings of the international conference on the applications of evolutionary computation (Part of EvoStar), Springer, pp 504–519
125. Lehman J, Stanley KO (2011) Abandoning objectives: evolution through the search for novelty alone. Evol Comput 19(2):189–223
126. Lehman J, Stanley KO (2008) Exploiting Open-endedness to solve problems through the search for novelty. In: Proceedings of the ALIFE
127. Risi S, Hughes CE, Stanley KO (2010) Evolving plastic neural networks with novelty search. Adapt Behav 18(6):470–491
128. Risi S, Vanderbleek SD, Hughes CE et al (2009) How novelty search escapes the deceptive trap of learning to learn. In: Proceedings of the 11th Annual conference on Genetic and evolutionary computation, pp 153-160
129. Conti E, Madhavan V, Such FP et al (2018) Improving exploration in evolution strategies for deep reinforcement learning via a population of novelty-seeking agents. In: Proceedings of the 32nd international conference on neural information processing systems (NIPS) pp 5032–5043
130. Reisinger J, Stanley K O, Miikkulainen R (2004) Evolving reusable neural modules. In: Proceedings of the genetic and evolutionary computation conference (GECCO), Springer, pp 69–81

131. Mouret J B, Doncieux S (2009) Evolving modular neural-networks through exaptation. In: Proceedings of the IEEE congress on evolutionary computation (CEC), IEEE, pp 1570–1577
132. Gomez F, Schmidhuber J, Miikkulainen R (2008) Accelerated neural evolution through cooperatively coevolved synapses. J Mach Learn Res 9:937–965
133. Praczyk T (2016) Cooperative co–evolutionary neural networks. J Intell Fuzzy Syst 30(5):2843–2858
134. Liang J, Meyerson E, Miikkulainen R (2018) Evolutionary architecture search for deep multitask networks. In: Proceedings of the genetic and evolutionary computation conference (GECCO), pp 466–473
135. Ellefsen KO, Mouret JB, Clune J (2015) Neural modularity helps organisms evolve to learn new skills without forgetting old skills. PLoS Comput Biol 11(4):e1004128
136. Velez R, Clune J (2017) Diffusion-based neuromodulation can eliminate catastrophic forgetting in simple neural networks. PLoS ONE 12(11):e0187736
137. Ellefsen KO, Huizinga J, Torresen J (2019) Guiding neuroevolution with structural objectives. Evol Comput 28(1):115–140
138. Knippenberg M V, Menkovski V, Consoli S (2019) Evolutionary construction of convolutional neural networks. arXiv preprint, arXiv:1903.01895
139. Liu P, El Basha MD, Li Y et al (2019) Deep evolutionary networks with expedited genetic algorithms for medical image denoising. Med Image Anal 54:306–315
140. Assunção F, Lourenço N, Machado P, et al (2019) Fast-DENSER++: evolving fully-trained deep artificial neural networks. arXiv preprint, arXiv:1905.02969
141. Li S, Sun Y, Yen GG, Zhang M (2021) Automatic design of convolutional neural network architectures under resource constraints. IEEE Trans Neural Netw Learn Syst. https://doi.org/10.1109/TNNLS.2021.3123105
142. Kyriakides G, Margaritis K (2022) Evolving graph convolutional networks for neural architecture search. Neural Comput Appl 34:899–909
143. Deng B, Yan J, Lin D (2017) Peephole: Predicting network performance before training. arXiv preprint arXiv:1712.03351
144. Istrate R, Scheidegger F, Mariani G, Nikolopoulos D, Bekas C, Malossi A C I (2019) Tapas: Train-less accuracy predictor for architecture search. In: Proceedings of the AAAI Conference on artificial intelligence 33: 3927–3934
145. Sun Y, Sun X, Fang Y, Yen GG, Liu Y (2021) A novel training protocol for performance predictors of evolutionary neural architecture search algorithms. IEEE Trans Evol Comput 25(3):524–536
146. Domhan T, Springenberg J T, Hutter F (2015) Speeding up automatic hyperparameter optimization of deep neural networks by extrapolation of learning curves. In: Proceedings of the Twenty-fourth International Joint Conference on artificial intelligence
147. Klein A, Falkner S, Springenberg TJ, Hutter F Learning curve prediction with Bayesian neural networks. In: Proceedings of the Fifth International Conference on learning representations, ICLR
148. Baker B, Gupta O, Raskar R, Naik N (2017) Accelerating neural architecture search using performance prediction. arXiv preprint arXiv:1705.10823
149. Zhang H, Jin Y, Jin Y, Hao K (2022) Evolutionary search for complete neural network architectures with partial weight sharing. IEEE Trans Evol Comput. https://doi.org/10.1109/TEVC.2022.3140855
150. Elsken T, Metzen J H, et al (2019) Efficient multi-objective neural architecture search via Lamarckian evolution. In: 7th International Conference on learning representations
151. Liang JZ, Miikkulainen R (2015) Evolutionary bilevel optimization for complex control tasks. In: Proceedings of the 2015 annual conference on genetic and evolutionary computation, pp 871–878
152. MacKay M, Vicol P, Lorraine J et al (2019) Self-tuning networks: Bilevel optimization of hyperparameters using structured best-response functions. arXiv preprint, arXiv:1903.03088
153. Sinha A, Malo P, Xu P et al (2014) A bilevel optimization approach to automated parameter tuning. In: Proceedings of the GECCO
154. Baldominos A, Saez Y, Isasi P (2018) Evolutionary convolutional neural networks: an application to handwriting recognition. Neurocomputing 283:38–52
155. Huang PC, Sentis L, Lehman J et al (2019) Tradeoffs in neuroevolutionary learning-based real-time robotic task design in the imprecise computation framework. ACM Trans Cyber-Phys Syst 3(2):14
156. Lipson H, Pollack JB (2000) Automatic design and manufacture of robotic lifeforms. Nature 406(6799):974
157. Durán-Rosal AM, Fernández JC, Casanova-Mateo C et al (2018) Efficient fog prediction with multi-objective evolutionary neural networks. Appl Soft Comput 70:347–358
158. Mason K, Duggan M, Barret E et al (2018) Predicting host CPU utilization in the cloud using evolutionary neural networks. Futur Gener Comput Syst 86:162–173
159. Khan GM, Arshad R (2016) Electricity peak load forecasting using CGP based neuro evolutionary techniques. Int J Comput Intell Syst 9(2):376–395
160. Grisci BI, Feltes BC, Dorn M (2019) Neuroevolution as a tool for microarray gene expression pattern identification in cancer research. J Biomed Inf 89:122–133
161. Abdikenov B, Iklassov Z, Sharipov A et al (2019) Analytics of heterogeneous breast cancer data using neuroevolution. IEEE Access 7:18050–18060
162. Wu Y, Tan H, Jiang Z, et al. (2019) ES-CTC: A deep neuroevolution model for cooperative intelligent freeway traffic control. arXiv preprint, arXiv:1905.04083
163. Trasnea B, Marina LA, Vasilcoi A, Pozna CR, Grigorescu SM (2019) GridSim: a vehicle kinematics engine for deep neuroevolutionary control in autonomous driving. In: Proceedings of the 2019 third IEEE international conference on robotic computing (IRC), IEEE, pp 443–444
164. Grigorescu S, Trasnea B, Marina L et al (2019) NeuroTrajectory: a neuroevolutionary approach to local state trajectory learning for autonomous vehicles. arXiv preprint, arXiv:1906.10971
165. Han F, Zhao MR, Zhang JM et al (2017) An improved incremental constructive single-hidden-layer feedforward networks for extreme learning machine based on particle swarm optimization. Neurocomputing 228:133–142
166. ElSaid A, Ororbia A, Desell T (2019) The ant swarm neuro-evolution procedure for optimizing recurrent networks. arXiv preprint, arXiv:1909.11849
167. Zhu W, Yeh WC, Chen J, Chen D, Li A, LinY (2019) Evolutionary convolutional neural networks using ABC. In: Proceedings of the 2019 11th international conference on machine learning and computing (ICMLC), pp 156–162