IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received September 1, 2019, accepted September 15, 2019, date of publication September 18, 2019,
date of current version September 30, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2942035

Correlation Analysis-Based Neural Network Self-Organizing Genetic Evolutionary Algorithm

ZENGHAO CHAI, XU YANG, ZHILIN LIU, YUNLIN LEI, WENHAO ZHENG, MENGYAO JI, AND JINFENG ZHAO

School of Computer Science and Technology, Beijing Institute of Technology, Beijing 100081, China

Corresponding author: Xu Yang (yangxu@tsinghua.edu.cn)

This work was supported in part by the National Natural Science Foundation of China under Grant 91846303, and in part by the National Natural Science Foundation of China under Grant 61502032.

ABSTRACT Recent years, there has been an ever increasing interest and investment on Artificial Intelligence (AI), both academic and industrial. As the hotspots in AI, Artificial Neural Networks (ANNs) have already been applied to a lot of different applications. However, traditional ANNs have disadvantages, such as fixed and redundant structure, resulting in requirement of large amount of training data and training time. Biological researches have shown that the biological neural network behaves in a more flexible way, with synapses building or withering according to requirement. In this paper, we present a Correlation Analysis Based Neural Network Self-Organizing Genetic Evolutionary Algorithm. Based on correlation analysis of training process, self-organizing combined with genetic evolutionary algorithm is applied to improve the performance efficiency and structural efficiency of the built neural network. Results show that our algorithm could generate neural networks with more compact structure and reasonable classification accuracy.

INDEX TERMS Artificial intelligence, neural network, correlation analysis, optimization, neural evolution.

I. INTRODUCTION

Artificial Neural Network (ANN) is a hotspot in artificial intelligence research. Deep learning Neural Network (DNN) is currently the main-stream artificial neural network. Traditional ANNs [1] have already been applied to many different tasks, and gain success. However, there are some disadvantages of traditional ANNs. For example, due to the characteristics of itself, BP neural network [2], [3] is easy to appear the phenomenon of local optimum, and might suffer from poor generalization. As for DNN, the large number of parameters is the guarantee of its accuracy, but the training time will become the nightmare of its users, and demanding huge mount of data for the purpose of fulfill the training process. It is found that the performance of neural networks will be improved in the process of structure complication, but this process will often be accompanied by a large number of redundancy in structure. The existence of a large number of redundancy in neural network structure is one of the main reasons for their falling into local optimum, and significantly prolongs the required training time. Therefore, it is necessary to design an algorithm that could wield the performance efficiency of neural networks while also enhances structural efficiency of neural network.

Mühlenbein [4] proposed the conjecture of the next generation neural network based on the analysis of the multilayer perceptron network and considered that the evolutionary neural network has a strong development potential. He believes that evolutionary neural networks will become a breakthrough point for the next generation of neural networks. Evolutionary Artificial Neural Network (EANN) [5] is a special kind of artificial neural network, which adaptively adjusts according to the change in external environment. Evolutionary Algorithms (EA) is the basis for realizing the adaptability of this kind of network. EANN simulates the evolution process of organisms in the natural environment by evolutionary algorithm. EANN is optimized by simulating the natural selection process of individuals in the biological evolution, by crossover and mutation.

Adaptability to the dynamic environment is the main feature of the EANN model, which is mainly reflected in three aspects: the weight of the connection, the network structure and the learning rules. Based on the three aspects on which EANN adaptation depends, the development of EANN [6] model can be divided into the following four types: 1. Evolve the weight alone, 2. Evolve the structure alone, 3. Evolve

The associate editor coordinating the review of this manuscript and approving it for publication was Xi Peng.

the learning rules alone, 4. Co-evolution of structure and weight.

The structure of a neural network represents the topology of neurons in that network. It describes how those neurons are connected from the point of view of spatial composition. And the weights of all the synapses in a neural network describes how those neurons are connected in view of information representation. Thus, in our opinion, the strong nonlinear mapping ability of neural network is the result of the synergy of structure and weight.

This paper's research focuses on the co-evolution of structure and weight [7]. The basis of co-evolution of structure and weight is to encode the structure and weight of neural network model, and then iterate through evolutionary algorithm to output the neural network model which conforms to the set problem.

In the field of EANN, the gradual complication of neural network tends to cause redundancy in structure, and sometimes the evolutionary algorithm might delete some potential individuals at the early phase of training mistakenly. To solve the above problems, a self-organizing evolutionary neural network algorithm based on the combination of genetic algorithm and correlation analysis is proposed in this paper. The contributions of our work are:

1) By simulating the construction process of biological neural network, Correlation Analysis Based Neural Network Self-organizing Algorithm (CANS), based on correlation analysis, is proposed to realize self-organization of neural network structure. In the process of neural network self-organization, the structure of the neural network is optimized and the redundancy in structure is avoided to a certain extent.
2) Based on the idea of evolutionary neural network and combining the output neural network of CANS with genetic algorithm [8], Correlation Analysis Based Neural Network Self-organizing Genetic Evolutionary Algorithm (GA-CANS) is proposed. It optimizes the initial network outputted by CANS through the evolutionary characteristics of genetic algorithm, and finally outputs the optimal individual obtained by the genetic algorithm iteration, which is the target neural network by decoding and restoring.

The following is organized as: Section 2 discusses related works; Section 3 introduces the CANS algorithm, while GA-CANS algorithm is described in Section 4. Experiment framework and results are given in Section 5. Finally, Section 6 contains the conclusion of this study.

II. RELATED WORKS

In 1996, Moriarty et al. proposed a new enhanced learning system called Symbiotic, Adaptive Neuro-Evolution, abbreviated as SANE [10]. SANE uses symbiotic evolution in a group of neurons connected to each other to form a complete neural network. SANE evolves a series of neurons rather than a complete network, which differs from other neuroevolutionary systems.

Gomez and Miikkulainen proposed a new neuroevolutionary system Enforced Sub-Populations (ESP) [11] in 1999, to solve the standard two-bar control task and the more difficult non-Markov version of the two-bar control problem. ESP is similar to SANE in that the population is composed of individual neurons rather than a complete network, and the subgroups of the neurons form a complete network. The ESP, however, assigns a separate population to each unit in the network, and a neuron can only be reassembled with members of its own subgroup. The population of neurons is divided into subgroups. The network is formed by randomly selecting a neuron from each subgroup. The main contribution of ESP is that it allows the network to evolve frequently.

Enhanced topology neural evolutionary NEAT [5] is a genetic algorithm used to generate evolutionary artificial neural networks proposed by Ken Stanley in 2002. It changes the weight parameters and structure of the network and tries to find a balance between the adaptability and diversity of evolutionary solutions. It proposes three key techniques: tracking genes with historical markers to allow the intersection of topologies, preserving innovation through the formation of new species, and progressively growing topologies from simple initial structures. NEAT attempts to evolve the weights and topology of neural networks simultaneously. In order to encode the network as a phenotype of genetic algorithms, NEAT uses a direct coding scheme, which means that each connection and neuron is clearly represented.

In 2014, Hyoung-uk et al. proposed a method called leap evolution adopted neural network (LEANN) [12] that optimizes the NN without prior knowledge such as the values of the variables and the structure of the NN for a given problem. Their method uses the GA plus other evolutionary methods inspired by nature or biology to find the optimal values of the perceptron variables and the structure, and could find an optimal structure and variables of the NN successfully for the XOR gate problem.

In 2015, a method [13] to reduce the storage and computation required by neural networks was proposed by Song Han et al. They focused on the important connections in neural networks, without affecting their accuracy at the same time. On the ImageNet dataset, their method reduced the number of parameters of AlexNet by one order of magnitude, without incurring accuracy loss.

In 2018, Sun *et al.* [14] have proposed a computationally economical algorithm for evolving unsupervised deep neural networks to efficiently learn meaningful representations, which is very suitable in the current Big Data era where sufficient labeled data for training is often expensive to acquire. A small proportion labeled data is utilized during evolution search to guarantee the learned representations to be meaningful.

III. CORRELATION ANALYSIS BASED NEURAL NETWORK SELF-ORGANIZING ALGORITHM

Traditional ANNs have some disadvantages. For example, BP algorithm with sigmoid activation function strictly

follows gradient descent learning, as a result, the gradient will approach zero when the output is close to 1 or 0; To achieve the desired effect of the model, large quantities of data for training is necessary, so DNNs are in lack of adaptability in solving small sample size problems; In the neural network model, the error function usually has several minimum points, so it is likely to fall into local optimum in the process of training [15].

When the scale of the network rises to a certain level, the parameters of the neural network will increase exponentially. Therefore, more parameters need to be trained and the dimension of the error function will inevitably get increased. As a result, the training time and the risk of falling into local optimum as well as the maximum points of the function get increased.

Therefore, consideration to guarantee of the accuracy and the reduction of parameter number in the neural network are two directions of the development of the neural network model.

In this section, based on the study of the construction process of biological neural network [16], Correlation Analysis Based Neural Network Self-organizing Algorithm , CANS is proposed. It complicates the neural network gradually during iteration, obtains the correlation of the connection in the neural network according to the training data, then realizes the optimization of the neural network structure, and finally outputs the neural network model that conforms to the target problem.

A. MOTIVATION

The inspiration of CANS originates from the construction process of biological neural network, that is, biological neural network will form structures with specific functions during its development. For biological neural network, those structures are the result of adaptations to the changes of external environment.

Professor Hebb argues that biological neural networks automatically reinforce connections between simultaneously activated neurons, and vice versa. As external stimuli occur, biological neural network automatically reinforces and gradually establishes connections between two simultaneously activated neurons. As a result, neural network with specific structure is initially formed. Then, with external stimuli, the structure will be optimized according to Hebb rules. Previously constructed connections will strengthen if the two neurons are simultaneously activated, and degenerate if not. Finally, a stable neural network structure is formed [17].

CANS is the guidance of neural network's gradual complication, designed to obtain neural networks with problem-solving ability. Compared with traditional neural networks, it's hoped that neural networks obtained by CANS have sparser structures with stronger generalization ability.

B. FLOW OF CANS

The flow of CANS is shown in Fig. 1. It can be divided into three phases: Initialization, Complication, and Output. There are some things need to be mentioned:

1) CANS can avoid problems in BP neural network such as falling into local optimum and gradient disappearance by optimizing the structure, and has good effect on small sample problems because of constant changes of the neural network structure.
2) In CANS, conditions for algorithm and fitting degree between structure and data set determine the ending of evolutionary process. The iteration of neural network structure can be completely handed by CANS automatically after running parameters and terminal conditions being set. Then specific neural network structure for given problem can be obtained.

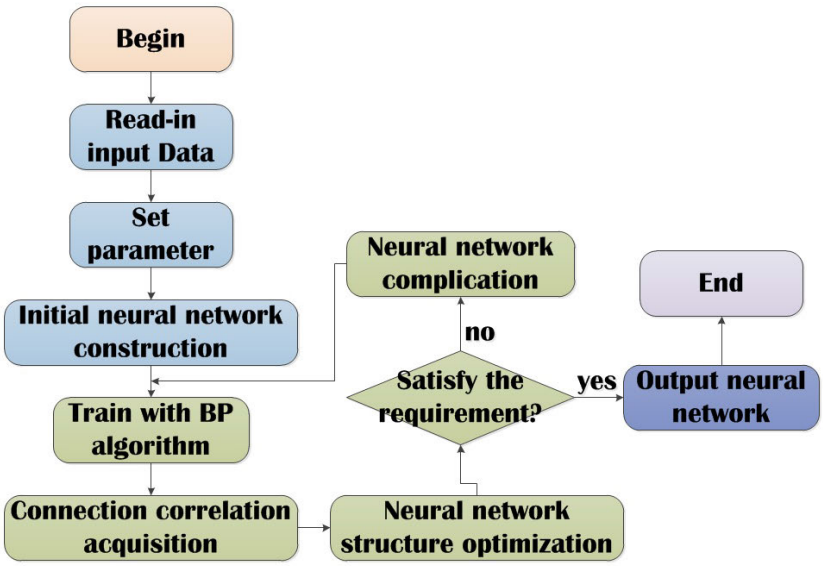


FIGURE 1. Flow of CANS algorithm.

3) The resulting neural network structure will be stratified according to the characteristics of neuronal connections, but the connections between neurons are produced in the iterative process, quietly different from commonly used layered neural networks.

1) INITIAL NEURAL NETWORK CONSTRUCTION

In initialization phase, CANS will analyze input data set, set related parameters according to the analysis, including neural number of each layer, and then construct initial neural network.

Initial neural network only contains two layers: input layer and output layer. Neurons between the two layers are fully connected with weights randomly set between 0 and 1, to simulate the process of signal transmission after stimulation of biological neurons.

In CANS, during constructions of initial neural network, new neurons are added into input layer and output layer in full connected manners, and new neurons are added the same way during complication of neural network.

2) TRAIN WITH BP LEARNING ALGORITHM

In this paper, BP learning algorithm [18] is introduced here to train the network. On the one hand, to optimize weights of connections and improve the accuracy of current neural network; on the other hand, to obtain the training log data that provides information for the next step of the algorithm.

The deviation between the actual output o and expected output d of neural network is the error of neural network. Suppose error is E, and l is the number of output neurons, then:

$$E = \frac{1}{2}(d-o)^2 = \frac{1}{2}\sum_{k=1}^{l}(d_k - o_k)^2 \tag{1}$$

If the above error definition formula is extended to hidden layer neurons, where m is the number of hidden layer neurons, y_j represents the output of the j^{th} hidden layer neurons, and w_{jk} represents the weight of the connection between the j^{th} hidden layer neuron and the k^{th} output neuron, then:

$$E = \frac{1}{2}\sum_{k=1}^{l}[d_k - f(net_k)]^2 = \frac{1}{2}\sum_{k=1}^{l}[d_k - f(\sum_{j=1}^{m} w_{jk}y_j)]^2 \tag{2}$$

If further extended to input neurons, where n is the number of input neurons, x_i represents the output of the i^{th} input neuron, and v_{ij} represents the weight of the connection between the i^{th} input neuron and the j^{th} hidden layer neuron, then:

$$\begin{aligned} E &= \frac{1}{2}\sum_{k=1}^{l}\{d_k - f[\sum_{j=1}^{m} w_{jk}f(net_j)]\}^2 \\ &= \frac{1}{2}\sum_{k=1}^{l}\{d_k - f[\sum_{j=1}^{m} w_{jk}f(\sum_{i=1}^{n} v_{ij}x_i)]\}^2 \end{aligned} \tag{3}$$

According to the above formula, the error of the network is the function of every connection weight value w_{jk}, v_{ij}, so the error E can be reduced by adjusting the weight value of connections.

3) CONNECTION CORRELATION ACQUISITION

In this paper, connection correlation refers to the importance and necessity of a connection in the current neural network relative to the whole neural network, and its acquisition refers to the process of obtaining the correlation of each connection by analyzing the training log after completing a round of training of neural network with BP algorithm. Training log records related information, including weight changes of each connection, and its input, output respectively.

Biologically, information transmission in neural networks is achieved by connections between neurons. When a neuron receives a stimulus from a portion of the upstream neuron and reaches a certain threshold, the neuron itself emits a spike signal to the downstream neurons, and then enters a cooling period [19]. According to Hebb rules, connections between two neurons will become weaker and weaker if they are not activated simultaneously. It can be assumed that during the cooling period, the neuron does not respond to a spike signal from its upstream neuron, so connection between them can be considered invalid. They are redundant relative to the entire neural network.

Training log analysis obtains the necessity of connections relative to neural network, which reflects the correlation between two connected neurons. The obtained correlation is used to provide guidance for optimization of connections in the network.

Connection correlation δ_i of a connection i is calculated as:

$$\delta_i = \begin{cases} \delta_i + +, & if\ conn\_in_t^i > k\ and\ conn\_out_t^i > k \\ \delta_i, & if\ conn\_in_t^i \le k\ or\ conn\_out_t^i \le k \end{cases} \tag{4}$$

Here, $conn\_in_t^i$ represents the membrane potential of the input neuron of connection i in time interval t, while $conn\_out_t^i$ represents the membrane potential of the output neuron of connection i in time interval t. t is a pre-defined time window. k is the threshold. If a neuron's membrane potential is higher than threshold, it would emit a spike signal.

According to Hebb rules, if both $conn\_in_t^i$ and $conn\_out_t^i$ are larger than k in a time interval t, it means that both the input neuron and the output neuron of connection i are activated, so in this time interval, connection i is transmitting useful information, thus connection i needs to be strengthened. Therefore, the connection correlation of connection i increases. Otherwise, it means that they are not both activated in time interval t, so connection i is not transmitting useful information, therefore, the connection correlation will not increase.

4) NEURAL NETWORK STRUCTURE OPTIMIZATION

Structure optimization is based on connection correlation, connections can be eliminated from the neural network if judged as unnecessary.

The necessity of a connection N_i is judged according to:

$$N_i = \begin{cases} 1, & if\ \delta_i > \alpha \cdot count \\ 0, & if\ \delta_i < \alpha \cdot count \end{cases} \tag{5}$$

Here, if N_i equals to 1, then connection i is treated as necessary; and if N_i is 0, then connection i is treated as unnecessary. α is a floating-point number between 0 and 1, representing the ratio of the number of simultaneous activations of two connected neurons to the total number of times the neural network adjusts during one iteration of the BP algorithm. *count* is the number of times the neural network adjusts during one iteration of the BP algorithm.

Actually, α also indicates the strength of structural optimization. In the process of structure optimization, the larger the α, the simpler the optimized neural network structure is, whereas the smaller the α is, the more complex the optimized neural network structure is.

However, because parameters α, δ and k influence each other, it is impossible to directly control the optimization stage by simply adjusting one parameter. To avoid potential problems, connections that considered to be related to current neural network are retained, that is to say, $\alpha = 1 - k$.

Structure optimization is actually the process of deleting invalid connections. The bases for deletion are as follows:

1) First, CANS algorithm refers to biological neural networks. In biological neural networks, when constructing a specific functional structure, the highly correlated neurons will follow synaptic plasticity and establish connections, while there will be no connection formed between neurons with weak correlation [20]. In initialization phase, neurons are fully-connected in the begin, it is necessary to delete invalid connections in optimization stage.
2) Second, BP learning algorithm is used to train, existing risks of local extreme state and excessive training time. Because connections are dimensions of the error function, the deletion of invalid connection will adjust the dimension. To a certain extent, the occurrence of local extremum and the phenomenon of gentle region will be reduced, or even avoided.
3) Third, the deletion of invalid connections also simplifies the structure of the outputted neural network, which not only improves the generalization ability of the outputted network to a certain extent, but also reduces the amount of data of the outputted network at run time. The speed of the neural network could also be improved.

5) NEURAL NETWORK COMPLICATION

Neural network complication refers to the process in which the algorithm dynamically adds neurons to the current neural network, according to the parameter n which is set at the initial stage of the operation of the algorithm. The premise of CANS algorithm executing neural network complicating process is: 1) after BP learning algorithm training and network structure optimization, the performance of current neural network cannot meet the output requirement; 2) the stop criterion is not meet.

In CANS, considering the iterations of the algorithm, the value range of n is $[n_{in}, n_{out}]$, where n_{in} and n_{out} correspond to the number of input and output neurons, respectively.

C. EVALUATION OF CANS

1) EVALUATION CONSTRAINTS

As we augured in the paper, traditional fixed structure ANNs suffer from redundancy in the neural network. This issue causes the prolonging of training time, huge amount of training data, and not energy efficient when used. We want to find a way to maintain the performance efficiency of neural network, while enhance the structural efficiency of it. Thus, the evaluation criteria we used in this paper are designed to evaluate if our algorithm has really done that.

a: PERFORMANCE EVALUATION

Accuracy and recall are two widely used metrics in machine learning to evaluate the quality of results [21]. Accuracy represents the proportion of the samples correctly classified to the total number of samples, while recall is the fraction of relevant instances that have been retrieved over the total amount of relevant instances. In multi-classification problems, there is a recall for each type of data.

b: STRUCTURE EVALUATION

In ideal condition, CANS algorithm not only requires a lower complexity of the output neural network structure, but also requires a balanced distribution of connections between non-input neurons and other neurons in the outputted neural networks. Networks with unbalanced connections are in lack of generalization ability, and unsuitable for the next iteration foundation. In order to evaluate the effect of the algorithm, it is necessary to design an evaluation coefficient to evaluate the rationality of its output network structure.

Considering the characteristics of the neural network model in ideal state, the structure evaluation function is constructed as follows:

$$x = \frac{num_{total}}{n} \tag{6}$$

$$y = \frac{num_{input}}{m} \tag{7}$$

$$\phi = \frac{\sum_{i=1}^{n}(num_i - x)^2}{n} + \frac{\sum_{j=1}^{m}(num_j - y)^2}{m} \tag{8}$$

where num_{total} is the total number of connections in neural network, n is the number of neurons other than input neurons, num_i is the number of input connections of the non-input neuron i, x is the average number of input connections of the non-input neurons in the model, num_{input} represents the total number of output connections of input neurons, m is the number of input neurons, num_j is the number of output connections owned by the *jth* input neuron, y is the average output number of input neurons. ϕ is the evaluation

coefficient of the neural network structure set in this paper. It is the sum of the variance of the number of output connections of each input neuron and the number of input connections of non-input neurons in the network.

The effect of output network structure is inversely proportional to the size of ϕ, that is, the smaller ϕ, the better the effect of output network structure, and vice versa.

c: OPTIMIZATION EVALUATION

Optimization evaluation evaluates the sparse degree of the optimized neural network compared with that without structural optimization. The optimization evaluation is defined as follows:

$$\omega = 1 - \frac{count_{now}}{count_{total}} \tag{9}$$

where $count_{now}$ is the number of connections that exist in the current neural network, $count_{total}$ is the total number of connections established during the whole iteration of the algorithm, ω is the evaluation coefficient of neural network optimization. The larger the ω, the more connections the neural network is deleted during the algorithm iteration, that is, the sparser the network structure is, and vice versa.

2) EVALUATION DATA SET

In this paper, CANS is used to classify the data sets to verify the validity and stability of the algorithm, and the output results of the algorithm are analyzed in order to propose an improved scheme. The experimental data is a standard data set from the UCI machine learning database, Nursery Data Set, with 12,960 samples. The attributes of this data set are shown in Tab. 1 [22], [23].

TABLE 1. Attributes of nursery data set.

Attribute	Value
health	recommended, priority, not_recom, very_recom
parents	usual, pretentious, great_pret
has_nurs	proper, less_proper, improper, critical, very_crit
form	complete, completed, incomplete, foster
children	1, 2, 3, more
housing	convenient, less_conv, critical
finance	convenient, inconv
social	non_prob, slightly_prob, problematic

In Tab. 1, one-hot encoding is adopted to encode discrete data in Nursery Data Set. The basic idea of one-hot encoding is to treat every value of discrete features as N states, only one of the N states has a state bit value of 1 and the rest of the state bit is 0.

Then, after one-hot encoding, the Nursery Data Set becomes 28 inputs and 4 outputs. In the experiment, totally disjoint training set and test set are used, and the ratio of the training set to the test set is 4: 1, and the order of the data in the sample is randomly scrambled.

3) EVALUATION RESULTS

Two experiments are designed. In experiment 1, the default parameters of the algorithm are used to verify the effect of the output network, by considering accuracy and recall. In experiment 2, the parameters related to the neural network structure of the algorithm are adjusted, and the stability of the algorithm is verified by comparing the experimental results with experiment 1.

a: EXPERIMENT 1

The parameters of experiment 1 are shown in Tab. 2.

TABLE 2. Parameter list of experiment 1.

Name	Definition	Value
count	Number of iteration	3
n	Number of added neurons in neural network complication process	10
n_{input}	Number of input neurons	28
n_{output}	Number of output neurons	4
k	Activation threshold of neurons	0.5
α	Threshold for connection deletion	0.5

Results for experiment 1 are shown in Tab. 3, and the output neural network is shown in Fig. 2.

TABLE 3. Results of experiment 1.

Evaluation constrains	Value
Accuracy	65%
Recall	70%, 68%, 60%, 62%
Optimization evaluation	0.2
Structure evaluation	0.4625

b: EXPERIMENT 2

The parameters of experiment 2 are show in Tab. 4.

TABLE 4. Parameter list of experiment 2.

Name	Definition	Value
count	Number of iteration	3
n	Number of added neurons in neural network complication process	10
n_{input}	Number of input neurons	28
n_{output}	Number of output neurons	4
k	Activation threshold of neurons	0.5
α	Threshold for connection deletion	0.7

On the basis of experiment 1, connection deletion threshold is added in experiment 2, to observe the effect of enhanced optimization on the stability and accuracy. Results for experiment 2 are shown in Tab. 5, and the output neural network is shown in Fig. 3.

TABLE 5. Results of experiment 2.

Evaluation constrains	Value
Accuracy	50%
Recall	55%, 43%, 42%, 60%
Optimization evaluation	0.5
Structure evaluation	0.9137

4) EVALUATION ANALYSIS

Experiment 1 shows that the neural network generated by CANS has the ability to classify the target data set, and the

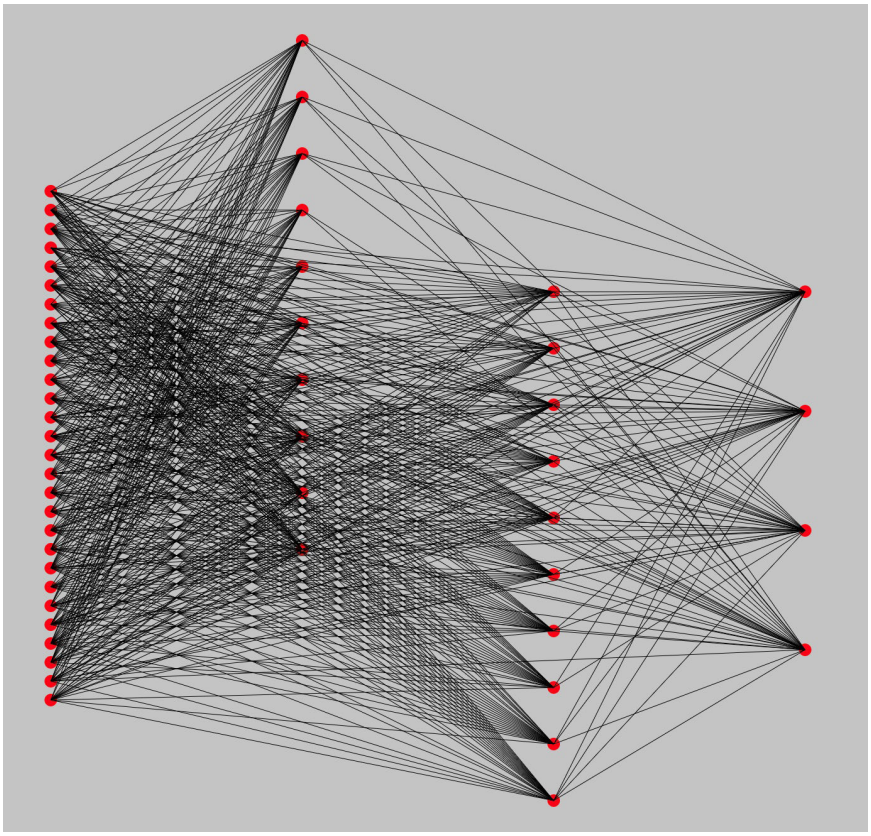


FIGURE 2. Output neural network for experiment 1.

number of connections is optimized compared with ANN of the same size. It is in line with the goal of algorithm design, that is, to simplify the neural network model by reducing the connections in the neural network. However, the outputted neural network is not up to the expectation of the algorithm design either in the aspect of neural network simplification or in the accuracy. It can be found from experiment 2 that although the parameters are adjusted to increase the simplified efficiency of the algorithm, the performance of the neural network outputted by the algorithm in experiment 2 is far less than that of the neural network outputted by experiment 1.

Results show that neural network outputted by CANS has certain functions, indicating that the algorithm is feasible and has the value of optimization and improvement.

After further study, on the basis of CANS and referring to the idea of evolutionary neural network [24], Correlation Analysis Based Neural Network Self-organizing Genetic Evolutionary Algorithm, GA-CANS is proposed in this paper.

IV. CORRELATION ANALYSIS BASED NEURAL NETWORK SELF-ORGANIZING GENETIC EVOLUTIONARY ALGORITHM

The optimizations of GA-CANS referring to CANS are:

1) Optimizing connection correlation based on Pearson correlation coefficient;
2) Optimizing invalid connection deletion method based on roulette method;
3) Optimizing the whole flow of algorithm based on genetic algorithm.

A. FLOW OF GA-CANS

The flow of GA-CANS is shown in Fig. 4. Compared with CANS, if requirement is met, the outputted neural network will be treated as a seed neural network to perform genetic algorithm for further optimization in GA-CANS while CANS outputs it as the final result directly.

B. OPTIMIZATION OF CONNECTION CORRELATION BASED ON PEARSON CORRELATION COEFFICIENT

According to analysis, the connection correlation acquisition method used in CANS is not compatible with the nature of BP training algorithm, which might explain the unideal behavior of CANS.

When BP algorithm trains and adjusts the weights of neural networks, each round of BP training is a continuous process. In other words, the adjustment of errors will affect the result of forward propagation in next training round, and the errors caused by this forward propagation. Therefore, the adjustment of connection weights between neurons is not only the adaptation of errors, but also affects the next round of connection weights adjustment process [25].

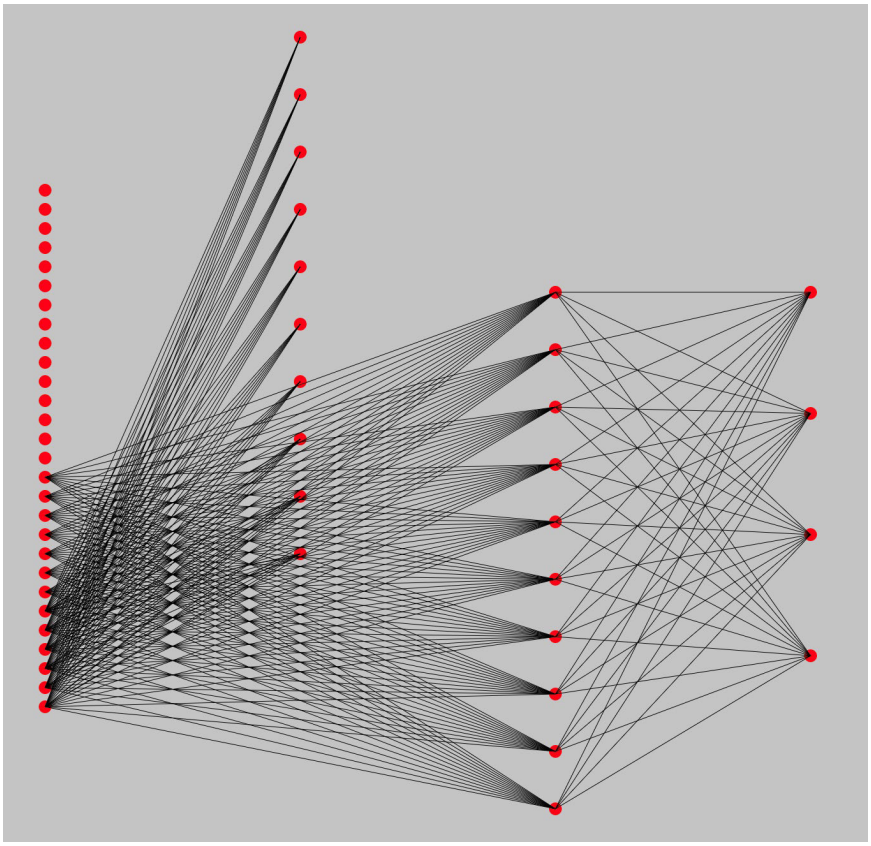


FIGURE 3. Output neural network for experiment 2.

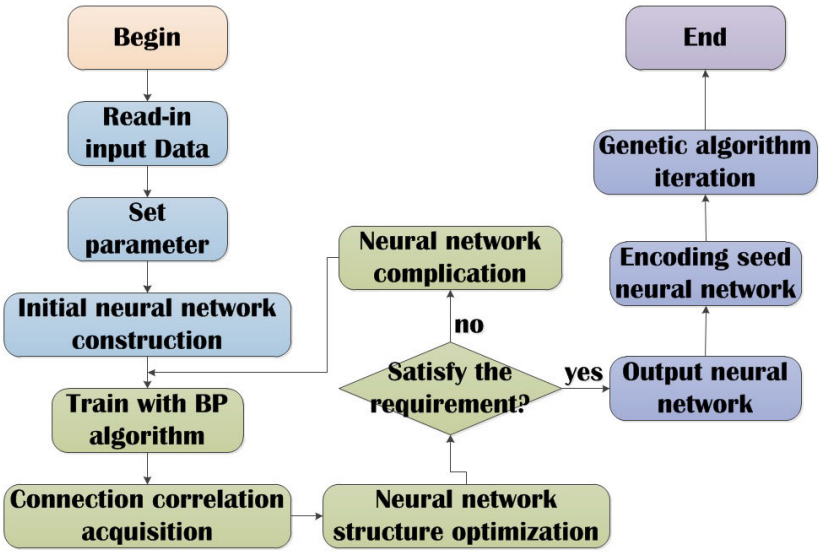


FIGURE 4. Flow of GA-CANS.

However, the connection correlation acquisition method used in CANS is only based on the one-round training results, which is too one-sided, completely ignoring the fact that the one-round training in the BP algorithm is only a link in a continuous process. At the same time, it can be seen from Eq. 4 that the connection correlation acquisition method used in CANS is only concerned with the forward propagation process in which the output of the connected input and output neurons is greater than the threshold value, which is bound to lose a lot of training information.

To avoid the above problems, Pearson correlation coefficient [26] is used to calculate the connection correlation in GA-CANS algorithm:

$$\gamma = \frac{\sum_{i=1}^{n}(X_i - \overline{X})(Y_i - \overline{Y})}{\sqrt{\sum_{i=1}^{n}(X_i - \overline{X})^2}\sqrt{\sum_{i=1}^{n}(Y_i - \overline{Y})^2}} \tag{10}$$

Pearson correlation coefficients range from -1 to 1. The greater the absolute value of Pearson correlation coefficient is, the stronger the correlation between the two is, and vice versa.

C. OPTIMIZATION OF INVALID CONNECTION DELETION METHOD BASED ON ROULETTE METHOD

The roulette method [27], which used in genetic algorithm to perform individual selection based on fitness, is chosen to implement as the invalid connection deletion mechanism in GA-CANS:

1) In genetic algorithm, roulette method only needs to obtain the proportion of selected individuals to screen the individuals in the population, and the determination of the proportion depends on the setting of parameters. Therefore, in GA-CANS algorithm, when using roulette method, only the sparse degree τ of the expected network need to be determined. τ denotes the proportion of connections reserved in the total number of connections in the network during iteration. The larger τ is, the more connections are reserved after iteration, and vice versa. The default value of τ will be determined through subsequent experiments.
2) The roulette method is based on predetermined proportion to complete the selection of connections, so the output neural network structure will show strong stability.
3) Because of the unique screening method of roulette method, not only the connection with strong correlation will be retained, but also some connections with weak correlation will be retained, thus providing certain diversity and ensuring the iterative potential of the network.

The roulette method is a common individual selection method in genetic algorithm, and it is also a random selection method, which is similar to roulette in gambling games. In genetic algorithm, when roulette method is called, the fitness of each individual is converted to the probability that the individual is selected. A disk represents all individuals, and then sectors are divided on the disk according to the selected probability value of each individual. The pointer simulates rotation, and the sector area in which the pointer stays after each rotation corresponds to the selected individual. Therefore, when using roulette method, the higher the degree of fitness, the greater the proportion of sectors in the disk, the more chances of being selected.

The formula of roulette method is:

$$p_i = F_i / \sum_{i=1}^{M} F_i \tag{11}$$

Here, p_i is the selection probability of each individual i, F_i is the fitness of that individual i, and M is the number of individuals.

The roulette method is an individual selection method with replacement, that is, the selected individuals can be selected again in the next screening process, which does not conform to the deletion of invalid connections in the GA-CANS algorithm. So the original roulette method needs to be modified. The modified roulette method for GA-CANS is as follows:

Algorithm 1 Modified Roulette Method for GA-CANS Algorithm

Input:
Neural network, N
Connection correlation value for all the connections
Stop criterion

Output:
Neural network after invalid connection deletion, N'

1: Read-in current neural network N;
2: **while** Stop criterion is not satisfied **do**
3: Acquire connection correlation value for each connection;
4: Set the inverse of the absolute value of the correlation of each connection as the fitness of the connection;
5: Perform roulette method to select connections;
6: Delete selected connections;
7: **end while**
8: Output optimized neural network N'.

The stop criterion is that the required sparse degree τ of the expected neural network has been met.

D. ENCODING OF SEED NEURAL NETWORK

In this paper, referring to the coding method of NEAT [5], GA-CANS uses double-stranded DNA to encode the structure and weights of neural network. An illustration of neural network encoding is shown in Fig. 5. The upper half of the graph is an example of the neural network structure. The lower half is the corresponding double-stranded DNA encoding. The upper half of the DNA is responsible for encoding the network structure and the lower half is responsible for encoding the weights. In the example of the neural network, the dashed line represents the deleted connection during the iteration of the algorithm, while the solid line represents the still retained connection that is in the current neural network.

In double stranded DNA coding, structure coding is the coding of neural network structure. It is a symbol encoding method that using 1 and 0 to represent establishment and disconnection of the connection respectively. The length of the structure coding is the same as the total number of connections established by the algorithm from the beginning of the initial stage to the final output network, that is, the structural coding of the neural network contains all the information of the establishment and disconnection of the connections, and each locus corresponds to a fixed connection. Weight coding

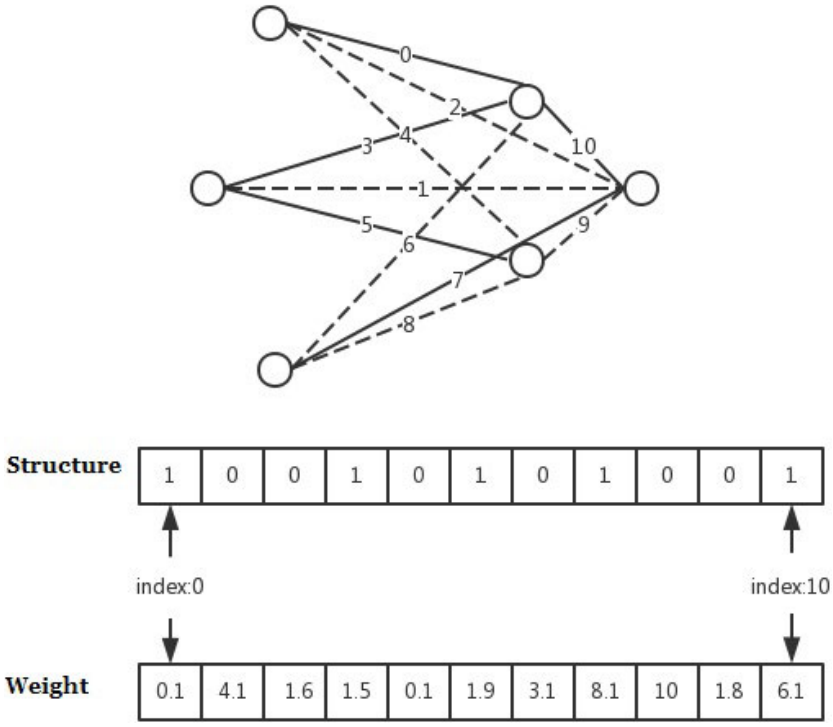


FIGURE 5. An example of neural network encoding in GA-CANS.

is the coding of weights of neural networks. It is a floating point coding method that contains the encoding of the weights of the existing connections as well as the weights of historical connections. It contains all the information in the process from the initial stage to the final output of the neural network, and each locus is fixed corresponding to the connection weight of a connection.

E. DESIGN OF GENETIC OPERATORS

Specific genetic operators are designed for GA-CANS algorithm.

1) SELECTION OPERATOR

In order to ensure the diversity of species in the population and the output of superior individuals, the roulette selection method and the optimal individual retention method are used as the selection operators simultaneously. The selection process of GA-CANS is as follow:

Step 1: Select the optimal individual in current population;
Step 2: Verify whether it is superior to the current optimal individual, if so, replace it;
Step 3: Choose the optimal individual enters the next generation;
Step 4: Select the remaining next generation individuals by roulette selection.

2) CROSSOVER OPERATOR

Double-stranded DNA is used to encode the neural network in GA-CANS, and the coding methods of the two DNA strands are completely different. It is necessary to design the corresponding crossover operator. The weight coding of neural network is based on the common floating point coding method, so the crossover operator can adopt the classical floating point coding crossover operator, which is not discussed here. However, the structure coding of neural networks requires the design of unique gene crossover operators due to the addition of loci and the complexity of the information contained therein.

The purpose of crossover in GA-CANS is to improve species diversity, optimize neural network structure, but don't need to complicate the network, which is different from NEAT algorithm, so it needs to be improved on this basis.

When performing a crossover operation, two matched DNA is selected at first, and then performs the crossover operation. When performing crossover, gene selection occurs only when the gene in the corresponding locus is different, otherwise the gene remains the same.

a: SELECTION OF MATCHED DNA

When selecting the matched DNA, the matching degree between each individual and other individuals should be calculated first, and then the two individuals with the highest matching degree should be selected to match. After the two selected individuals have completed the crossover operation, the other two individuals with the highest matching degree are selected from the remaining individuals.

The formula for calculating the matching degree between individuals is as follows:

$$f(x_i^1, x_i^2) = \begin{cases} 1, & x_i^1 = x_i^2 \\ 0, & x_i^1 \neq x_i^2 \end{cases} \tag{12}$$

$$\rho = \sum_{i=1}^{l} \left[f(x_i^1, x_i^2) + \frac{1}{(w_i^1 - w_i^2)^2} \right] \tag{13}$$

Here x_i^1 is the coding information in the *ith* locus in the structural coding of individual 1, x_i^2 is the coding information in the *ith* locus in the structural coding of individual 2, w_i^1 is the coding information in the *ith* locus in the weight coding of individual 1, w_i^2 is the coding information in the *ith* locus in the weight coding of individual 2, ρ is the match degree between individual 1 and 2.

b: GENE SELECTION

Roulette selection is served as the gene selection method, that is, if the fitness of the individual in which the gene is located is relatively high, the probability of being selected is higher, and the probability of being selected on the contrary is lower.

3) MUTATION OPERATOR

In the algorithm, since the symbol code is used in the structure coding, the mutation operator adopts uniform mutation, while Gaussian mutation is adopted as the mutation operator due to the floating-point coding in the weight coding. Then, on the basis of Gaussian mutation, BP algorithm is used to train reserved connection weights in order to speed up the convergence of weights and improve the fitness of individuals. The individual mutation operator operates as follows:

Step 1: Traverse each locus in individual structural coding;

Step 2: Generate a floating point number between 0 and 1 randomly. If the floating point number is less than the preset mutation probability p, the current corresponding structure encoding is reversed, otherwise it remains the same.

Step 3: Perform Gaussian mutation for weight coding.

F. TIME COMPLEXITY ANALYSIS OF GA-CANS

Regardless of the complexity of the activation function, in initial neural network, the number of neurons in input, hidden, output layers are $n1$, $N2$, and $n3$ respectively. Then the complexity of BP algorithm in training a single sample once is as follows:

$$\Theta(N_2(n_1 + n_3)) \tag{14}$$

If the sample size is m and the iterations is T, the complexity of the training algorithm is as follows:

$$\Theta(mTN_2(n_1 + n_3)) \tag{15}$$

BP algorithm is used to train in GA-CANS. In initial neural network, the number of neurons in input, hidden, output layers are n_1, n_2, and n_3 respectively, and the neurons added in hidden layer of each round are Δn_2, the sparse degree is τ. The training is carried out in n rounds and t iterations per round. The number of neurons in hidden layer in the k_th round training is:

$$\tau^{k-1}(n_2 + (k-1)\Delta n_2) + (k-1)\Delta n_2 \tag{16}$$

So the time complexity of the k_th round training is as follows:

$$\Theta(mt\{[\tau^{k-1}(\Delta n_2 + (k-1)\Delta n_2) + (k-1)n_2](n_1 + n_3)\}) \tag{17}$$

The total time complexity of n rounds is:

$$\Theta(\sum_{k=1}^{n} \{mt[\tau^{k-1}(n_2 + (k-1)\Delta n_2) + (k-1)\Delta n_2](n_1 + n_3)\}) \tag{18}$$

In this paper, the initial number of neurons of hidden layer n_2 is 0. If the sparse degree is default to 0.5, the complexity of GA-CANS is approximately expressed as follows:

$$\Theta(mt(n_1 + n_3)\Delta n_2[\frac{n(n-1)}{2} + \frac{(n-1)\tau^{n+1} - n\tau^n + \tau}{(1-\tau)^2}]) \tag{19}$$

When the number of training rounds is large enough, the above formula can be approximately estimated by neglecting the primary term and the constant term.

$$\Theta(mt(n_1 + n_3)\Delta n_2 \frac{n(n-1)}{2}) \tag{20}$$

If iterations $T = nt$ and the total hidden layer neurons $N_2 = n\Delta n_2$, when n is large enough,then:

$$\Theta(mT(n_1 + n_3)N_2) > \Theta(mt(n_1 + n_3)\Delta n_2 \frac{n(n-1)}{2}) \tag{21}$$

As a result, in theory, the time complexity of GA-CANS is less than that of BP algorithm.

V. EXPERIMENTS

A. EXPERIMENTAL FRAMEWORK

In order to further improve the effect of the algorithm, a visual neural network research platform is designed to have a better understand of the algorithm and the influence mechanism of its parameters on the evolution of network structure.

The visual research platform designed in this paper is a web application. The web server loads the log file output from the algorithm executed in the background, then parses the obtained file to obtain the corresponding neural network structure information. Furthermore, the visualization of neural network structure is realized by using Snap to construct SVG image in the front page. The front-end pages of the platform can also control the loading of different generations of neural network structure data, and then realize the conversion between different iterations of the same neural network and the visualization of the network structure.

The visualization research platform designed in this paper includes three modules: log analysis module, network structure visualization module, and evolutionary iterative control module:

1) Log analysis module: The outputted data in the course of training and structure evolution is a certain kind of structured data. Therefore, the log outputted by the algorithm should be a log file that holds structured data. The function of this module is to analyze the structured log data, obtain the relevant information, and then transfer it to the network structure visualization module for display.
2) Network structure visualization module: The function of this module is to construct the structure diagram of neural network and visualize display on the basis of the information obtained by log analysis.
3) Evolutionary iterative control module: The function of this module is to select different parameters to obtain the structure data of different generations of neural networks, and then to show and compare the structure diagrams of different generations of neural networks.

B. DATA SETS

Three data sets are selected to evaluate our algorithm.

1) NURSERY DATA SET

The first data set is a standard dataset Nursery Data Set from UCI machine learning database. It was derived from a hierarchical decision model originally developed to rank applications for nursery schools. The basic information about this data set is shown in Tab. 6. Detailed attributes and categories have discussed in Section III-C2.

TABLE 6. Basic information of nursery data set.

Number of samples	Number of attributes	Number of categories
12,960	8	4

2) ADULT DATA SET

The second data set is a standard dataset Adult Data Set from the UCI machine learning database [28]. It is used to predict whether a citizen's annual income exceeds $50,000 a year based on census data. The data set is a clean set of data sources extracted by Barry Becker from the 1994 census database , including 48842 samples.

The basic information about this data set is shown in Tab. 7. Detailed attributes and categories are shown in Tab. 8. After the analysis of the data set, one-hot method is also used to encode the data set. After processing, the input feature of the dataset is 81 and the output feature is 2 respectively, which are the corresponding number of input and output neurons in the iterative phase.

3) 6-XOR_64BIT DATA SET

The third data set is a standard dataset 6-XOR_64bit Data Set from the UCI machine learning database [29]. This dataset is generated using 6-XOR arbiters of 64bit stages PUF,

TABLE 7. Basic information of adult data set.

Number of samples	Number of attributes	Number of categories
48,842	14	2

TABLE 8. Attributes of adult data set.

Attribute	Value
age	Continuous
workclass	Private, Self-emp-not-inc, Self-emp-inc, Federal-gov, Local-gov, State-gov, Without-pay, Never-worked
fnlwgt	Continuous
education	Bachelors, Some-college, 11th, HS-grad, Prof-school, Assoc-acdm, Assoc-voc, 9th, 7th-8th, 12th, Masters, 1st-4th, 10th, Doctorate, 5th-6th, Preschool
education-num	Continuous
marital-status	Married-civ-spouse, Divorced, Never-married, Separated, Widowed, Married-spouse-absent, Married-AF-spouse
occupation	Tech-support, Craft-repair, Other-service, Sales, Exec-managerial, Prof-specialty, Handlers-cleaners, Machine-op-inspct, Adm-clerical, Farming-fishing, Transport-moving, Priv-house-serv, Protective-serv, Armed-Forces
relationship	Wife, Own-child, Husband, Not-in-family, Other-relative, Unmarried
race	White, Asian-Pac-Islander, Amer-Indian-Eskimo, Other, Black
sex	Female, Male
capital-gain	Continuous
capital-loss	Continuous
hours-per-week	Continuous
native-country	United-States, Cambodia, England, Puerto-Rico, Canada, Germany, Outlying-US(Guam-USVI-etc), India, Japan, Greece, South, China, Cuba, Iran, Honduras, Philippines, Italy, Poland, Jamaica, Vietnam, Mexico, Portugal, Ireland, France, Dominican-Republic, Laos, Ecuador, Taiwan, Haiti, Columbia, Hungary, Guatemala, Nicaragua, Scotland, Thailand, Yugoslavia, El-Salvador, Trinadad&Tobago, Peru, Hong, Holand-Netherlands

which consists of 2.4 million rows and 65 attributes. The range of each attributes is 1 or −1, and the last attribute is the class label. It is divided into two sets: training set (2 million) and testing set (400K). The basic information about this data set is shown in Tab. 9.

TABLE 9. Basic information of 6-XOR_64bit data set.

Number of samples	Number of attributes	Number of categories
2,400,000	64	2

C. RESULTS AND ANALYSIS

1) IMPROVEMENT OF PEARSON CORRELATION COEFFICIENT ON CONNECTION CORRELATION ACQUISITION

To evaluate the effect on improving connection correlation acquisition, the previous connection correlation acquisition in

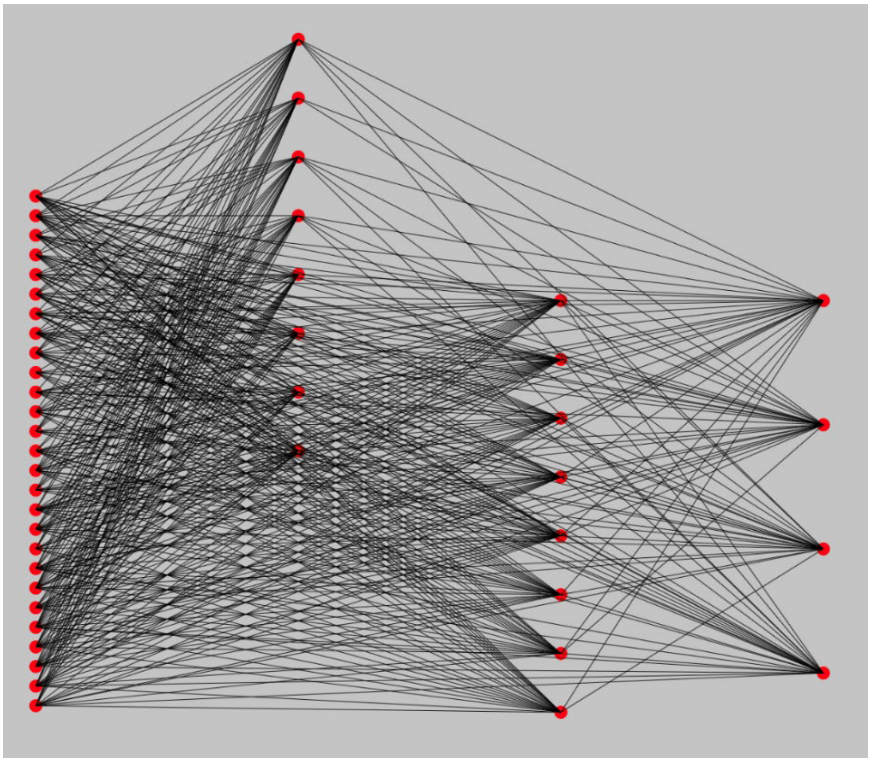


FIGURE 6. Result after person correlation coefficient improvement.

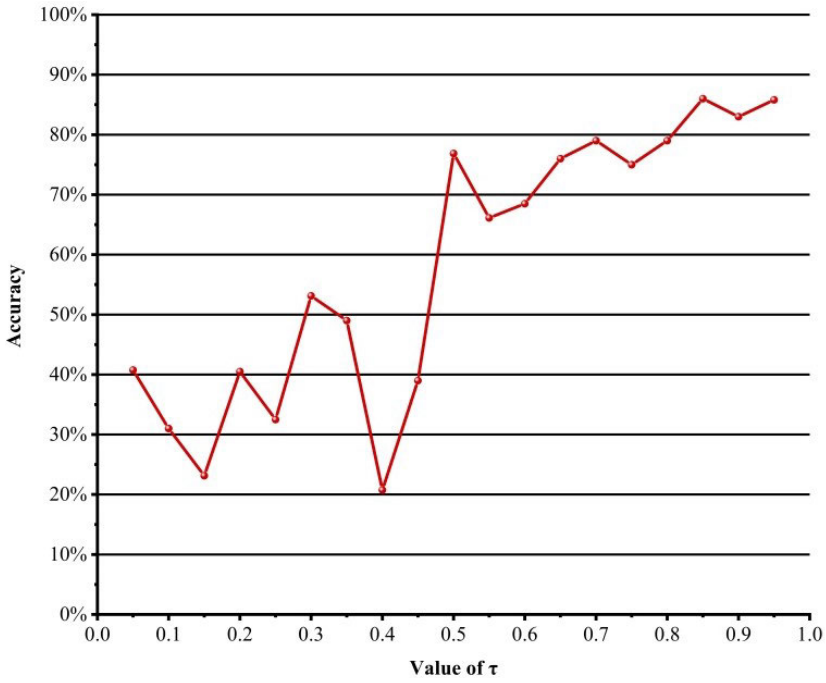


FIGURE 7. Influence of parameter τ on accuracy.

CANS is replaced by the method based on Pearson correlation coefficient. The data set used is Nursery Data Set. Setting of other parameters is the same as Tab. 2. The results are shown in Tab. 10, while the result of basic CANS algorithm has shown in Tab. 3 before.

The result neural network is shown in Fig. 6.

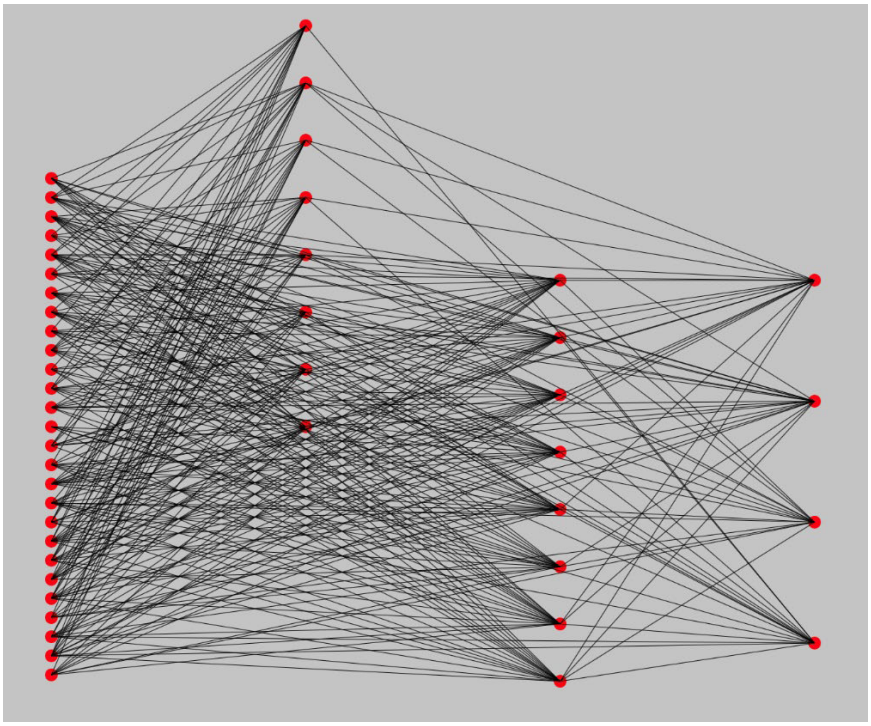


FIGURE 8. Result after roulette method improvement.

TABLE 10. Improvement of pearson correlation coefficient on connection correlation acquisition.

Evaluation constrains	Value
Accuracy	76%
Recall	74%, 80%, 78%, 72%
Optimization evaluation	0.2162
Structure evaluation	0.4525

Experimental comparison shows that, after using Pearson correlation coefficient instead, the accuracy is obviously improved. In several subsequent experiments of the same type, the effect of the algorithm remains stable. Thus, it is feasible to improve the connection correlation acquisition by using Pearson correlation coefficient.

2) IMPROVEMENT OF ROULETTE METHOD ON INVALID CONNECTION DELETION

To evaluate the effect of roulette method, the invalid connection deletion mechanism in the improved CANS, gotten in Section V-C1, is replaced by roulette method. The data set used is Nursery Data Set. Setting of other parameters is the same as Tab. 2.

When using roulette method, τ, the sparse degree of output network, should be determined, which denotes the proportion of connections reserved in the total number of connections in the network during iteration. The larger τ is, the more connections are reserved after iteration, and vice versa.

Fig. 7 shows the great effect of value τ on accuracy. The parameter τ and the accuracy of the final output model generally show a positive correlation trend. However, when τ is greater than 0.5, the accuracy of output network increases slowly. So the value of τ is set to 0.5 in this research. But we should mention that, the value of τ could be adjusted according to the actual performance requirement of the algorithm, that is, the model with higher accuracy can be obtained by increasing the value of τ.

The results are shown in Tab. 11 when setting τ to 0.5.

TABLE 11. Improvement of roulette method on invalid connection deletion.

Evaluation constrains	Value
Accuracy	84.25%
Recall	84%, 80%, 86%, 78%
Optimization evaluation	0.4563
Structure evaluation	0.325

The result neural network is shown in Fig. 8.

The neural network outputted by the algorithm makes progress in accuracy and is more stable in structure evaluation. In several subsequent experiments of the same type, results show that the effect of structure evaluation will also affect the accuracy of the final output of the algorithm.

3) COMPARISON WITH OTHER METHODS

Six indicators are selected to compare with other algorithms: Accuracy, Optimization evaluation, Structure evaluation, Iterations, Loss, and Relative training time.

Accuracy represents the performance of the neural network generated by the algorithm, and the higher Accuracy, the better performance of the neural network.

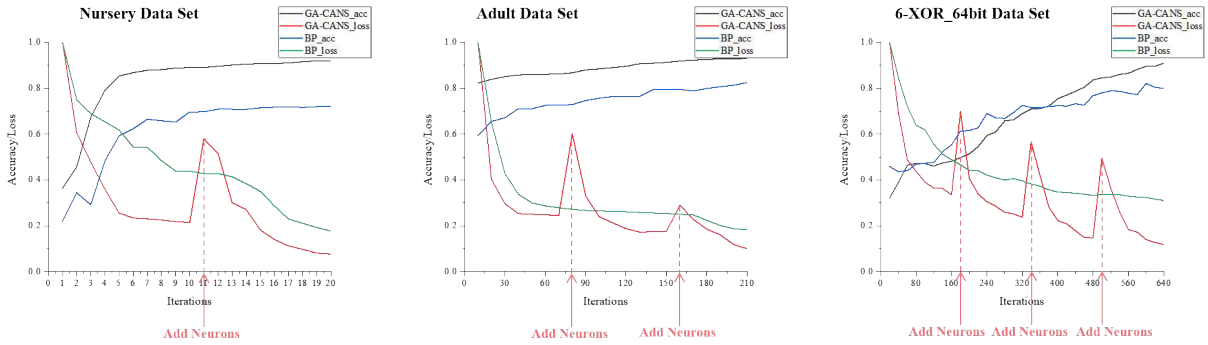


FIGURE 9. The comparison between GA-CANS and BP algorithm.

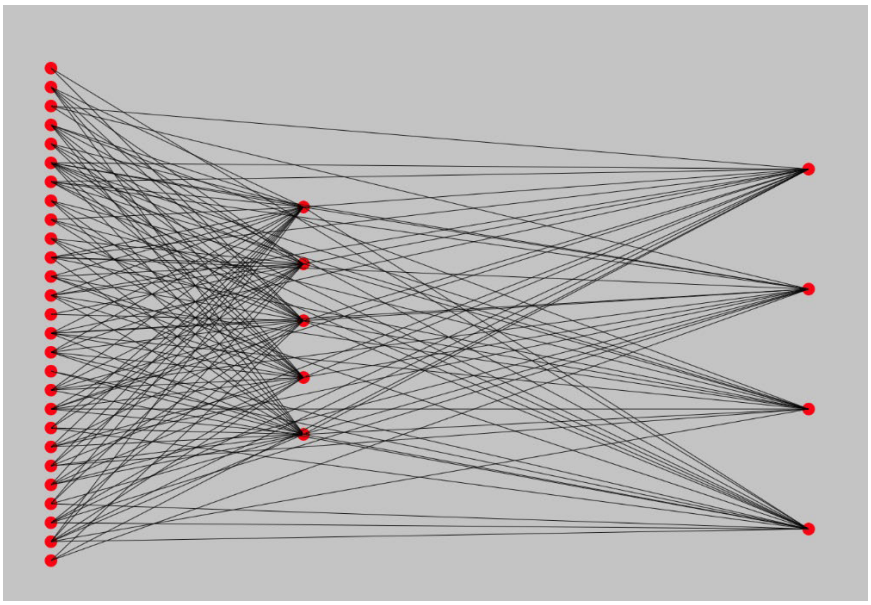


FIGURE 10. Output neural network for nursery data set.

Optimization evaluation represents the network optimization degree during the iteration stage of the algorithm. The higher Optimization evaluation is, the simpler the output network is, and vice versa.

Structure evaluation is unique to the algorithm designed in this paper. It represents the rationality of the neural network outputted by the algorithm. The lower Structure evaluation is, the higher rationality of the neural network outputted by the algorithm.

Iterations refers to the number of iterations before the algorithm outputs the optimal model. The fewer iterations, the faster output of the optimal model is.

Loss is used to evaluate the model. The smaller the loss, the better performance of the model. Cross Entropy loss function is used in this paper.

Training time is the cost of time during training, and in this paper, relative training time is used compared with auto-sklearn. The less training time, the better performance of training algorithm.

The comparison between GA-CANS and BP algorithm in three data sets is shown in Fig. 9.

The comparison of results on Nursery Data Set are shown in Tab. 12. The output neural network by GA-CANS is shown in Fig. 10.

The comparison of results on Adult Data Set are shown in Tab. 13. The output neural network by GA-CANS is shown in Fig. 11.

The comparison of results on 6-XOR_64bit Data Set are shown in Tab. 14. The output neural network by GA-CANS is shown in Fig. 12.

When setting BP algorithm and NEAT algorithm have the same iterations as GA-CANS, the accuracy of each data set is shown in Tab. 15, respectively.

The comparison of relative training time shown in Tab. 12, 13, 14 proves that GA-CANS algorithm has lower time complexity. Especially in large scale data set, the training time is obviously less compared with other three methods. Meanwhile, Fig. 9 shows the accuracy-iterations and

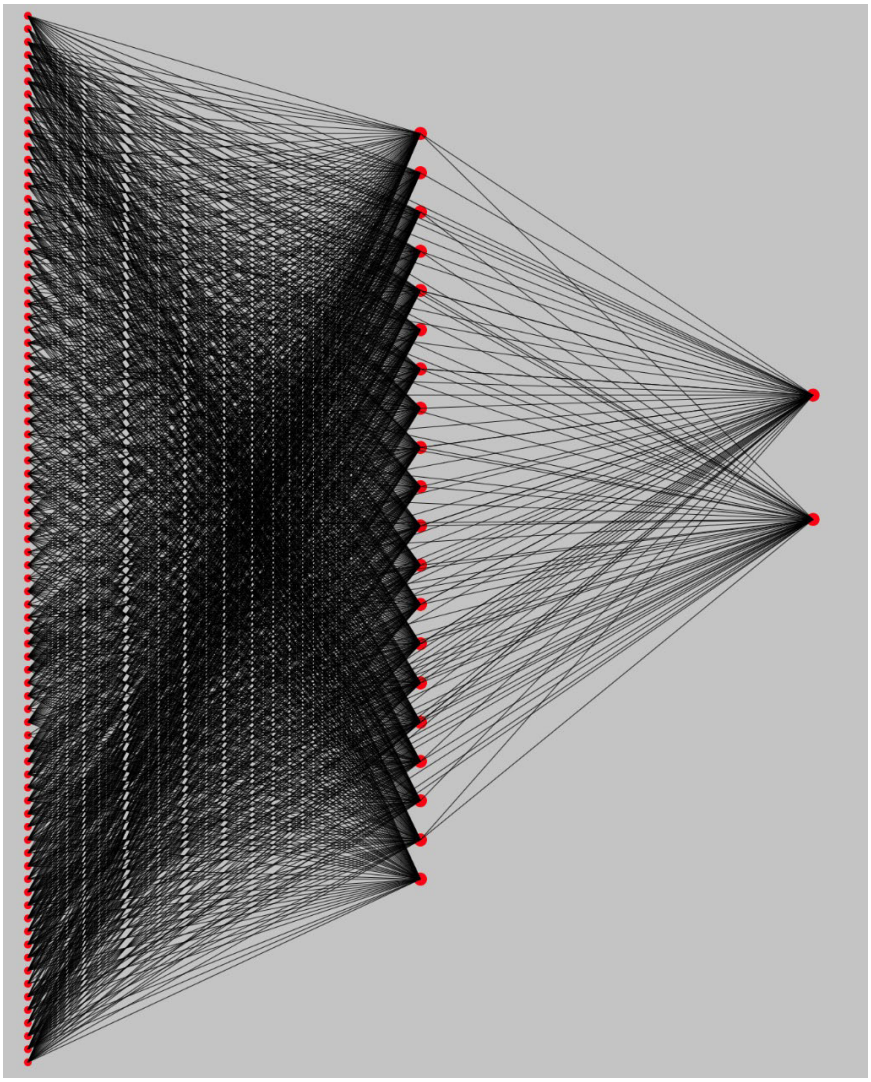


FIGURE 11. Output neural network for adult data set.

TABLE 12. Comparison of result on nursery data set.

Algorithm	Accuracy	Optimization evaluation	Structure evaluation	Iterations	Loss	Relative training time
GA-CANS	92.0%	0.5163	0.2025	20	0.076	0.19
NEAT	97.6%	0.5534	2.3949	92	0.096	0.21
BP neural network	98.6%	0	6.1728	100	0.105	0.19
auto-sklearn	98.8%	-	-	-	0.043	1

TABLE 13. Comparison of result on adult data set.

Algorithm	Accuracy	Optimization evaluation	Structure evaluation	Iterations	Loss	Relative training time
GA-CANS	93.0%	0.3065	0.2325	210	0.099	0.23
NEAT	94.6%	0.3284	2.5819	300	0.104	0.26
BP neural network	97.6%	0	13.888	710	0.136	0.27
auto-sklearn	97.5%	-	-	-	0.081	1

loss-iterations curves of GA-CANS algorithm and BP algorithm at iterations of 20, 210 and 640 in three data sets respectively. The peak of GA-CANS loss curve is caused by adding neurons dynamically in the training process. The performance of GA-CANS algorithm is stable in different scales of data set. Before convergence, the accuracy and loss of

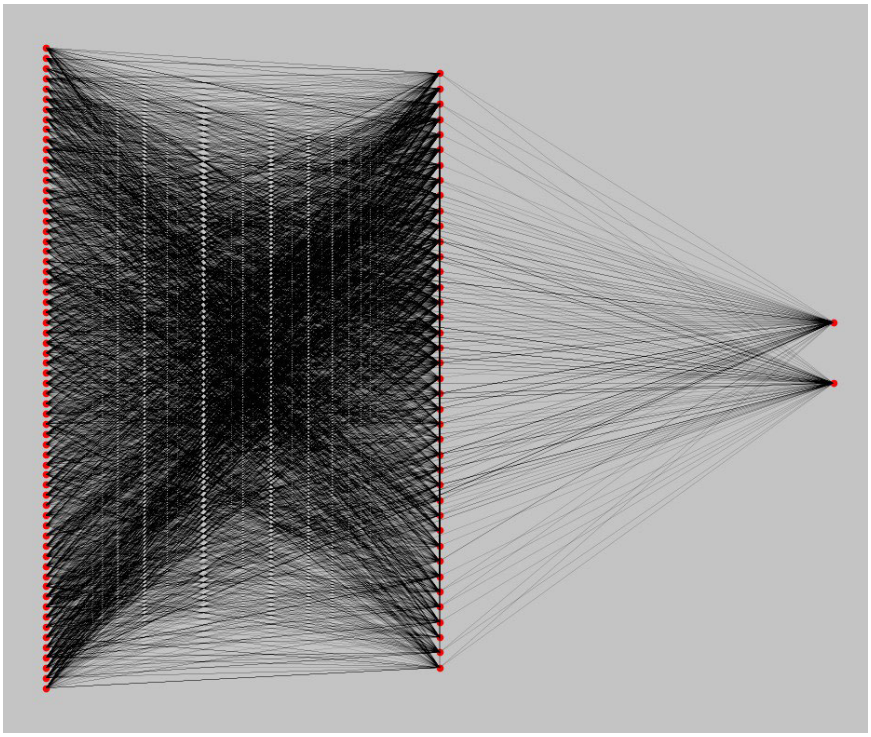


FIGURE 12. Output neural network for 6-XOR_64bit data set.

TABLE 14. Comparison of result on 6-XOR_64bit data set.

Algorithm	Accuracy	Optimization	Structure	Iterations	Loss	Relative training time
GA-CANS	90.9%	0.6719	0.4443	640	0.118	0.40
NEAT	91.6%	0.6398	2.3597	1020	0.119	0.60
BP neural network	92.9%	0	10.8592	1900	0.107	0.58
auto-sklearn	95.6%	-	-	-	0.104	1

TABLE 15. Accuracy of three data set under the same iterations.

Data Set	GA-CANS	NEAT	BP neural network
Nursery Data Set	92.0%	86.3%	72.5%
Adult Data Set	93.0%	87.7%	82.5%
6-XOR_64bit Data Set	90.0%	82.1%	80.3%

GA-CANS algorithm are better than BP algorithm, with the less loss and higher accuracy compared when the two algorithms have same iterations before GA-CANS convergence.

Experimental results show that GA-CANS with evolutionary algorithm is far superior to CANS in terms of accuracy and structure evaluation. GA-CANS can converge with fewer iterations, and the accuracy of outputted model can reach more than 90%. In addition, compared with BP algorithm, NEAT algorithm and auto-sklearn [31], although GA-CANS is slightly worse in accuracy, it is better than BP algorithm and NEAT algorithm in structure optimization and iterations, and its training time is much shorter than auto-sklearn. GA-CANS optimizes the construction of initial network, and the optimal neural network can be generated with less iterations and training time.

The performance of BP algorithm and NEAT algorithm reduce sharply when iterations is the same as that of GA-CANS. In addition, one kind of AutoML [30], auto-sklearn, is used to compare with GA-CANS, which takes a long time to get the optimal model. When the training time of auto-sklearn is reduced to the same as that of GA-CANS, the appropriate network model cannot be obtained in a short time, so the accuracy reduces rapidly.

VI. CONCLUSION

Two algorithms are proposed in this paper. In CANS, acquiring connection correlation information through analyzing training data is regard as the basic idea to optimize the structure of neural network and is implemented. Then generating GA-CANS algorithm by combing genetic algorithm with CANS. Several data sets are evaluated by the proposed algorithms and other state-of-the-art algorithms.

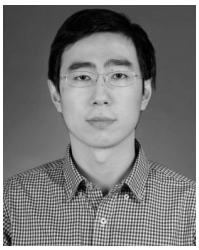
Using data set with different scales, results show that GA-CANS can be used to study classification problems in the field of artificial intelligence. It performs well in structure optimization and weight optimization, decreases iterations and training time. However, the accuracy of GA-CANS is

not as well as the other three algorithms, it might due to the over optimization of the structure of the neural network. In further study, detailed analysis should be applied. In this paper, Pearson correlation coefficient is used to describe the relationship between neurons, which might limit the function of the model merely using linear relationship, further research should focus on the relationship between neurons to improve the accuracy.

REFERENCES

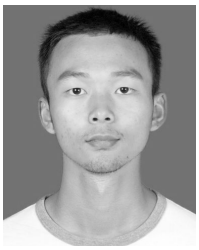
[1] A. M. Saxe, J. L. Mcclelland, and S. Ganguli, "Exact solutions to the nonlinear dynamics of learning in deep linear neural networks," in *Proc. ICLR*, 2013, pp. 1–22.
[2] J. Li, J.-H. Cheng, J.-Y. Shi, and F. Huang, "Brief introduction of back propagation (BP) neural network algorithm and its improvement," in *Advances in Computer Science and Information Engineering*. Berlin, Germany: Springer, 2012.
[3] Z. Qian *et al.*, "Advances in computer science and information engineering," *Lect. Notes Elect. Eng.*, vol. 5, no. 9, p. 24, 2012.
[4] H. Mühlenbein, "Limitations of multi-layer perceptron networks—Steps towards genetic neural networks," *Parallel Comput.*, vol. 14, no. 3, pp. 249–260, 1990.
[5] K. O. Stanley and R. Miikkulainen, "Evolving neural networks through augmenting topologies," *Evol. Comput.*, vol. 10, no. 2, pp. 99–127, 2002.
[6] W. Yao, Q. Wan, Z. Chen, and J. Wang, "The researching overview of evolutionary neural networks," *Comput. Sci.*, vol. 31, no. 3, pp. 125–129, 2004.
[7] D. C. Perera, M. S. S. Mathotaarachchi, L. Udawatta, and A. S. Perera, "ANNEbot: An evolutionary artificial neural network framework," in *Proc. Int. Conf. Intell. Adv. Syst.*, Jun. 2012, pp. 40–45.
[8] A. M. Andrew, "Systems: An introductory analysis with applications to biology, control, and artificial intelligence, by John H. Holland," *Robotica*, vol. 11, no. 5, p. 489, 1993.
[9] M. Matteucci, "EleaRNT: Evolutionary learning of rich neural network topologies," Dept. Comput. Sci., Carnegie Mellon Univ., Pittsburgh PA, USA, Tech. Rep. N CMU-CALD-02-103, 2002.
[10] D. E. Moriarty and R. Miikkulainen, "Efficient reinforcement learning through symbiotic evolution," *Mach. Learn.*, vol. 22, nos. 1–3, pp. 11–32, 1996.
[11] F. J. Gomez and R. Miikkulainen, "Solving non-Markovian control tasks with neuroevolution," in *Proc. 16th Int. Joint Conf. Artif. Intell.* Burlington, MA, USA: Morgan Kaufmann Publishers, 1999, pp. 1356–1361.
[12] H. Hyoung-Uk and J.-K. Kim, "An evolutionary genetic neural networks for problems without prior knowledge," in *Proc. Int. Conf. Natural Comput.*, Aug. 2014, pp. 1–6.
[13] S. Han, J. Pool, J. Tran, and W. Dally, "Learning both weights and connections for efficient neural network," in *Proc. 29th Annu. Conf. Neural Inf. Process. Syst.*, 2015, pp. 1135–1143.
[14] Y. Sun, G. G. Yen, and Z. Yi, "Evolving unsupervised deep neural networks for learning meaningful representations," *IEEE Trans. Evol. Comput.*, vol. 23, no. 1, pp. 89–103, Feb. 2019.
[15] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, p. 436, 2015.
[16] S. Brenner, "Principles of neural network construction," in *Neurosciences and Ethics*. Berlin, Germany: Springer, 1988.
[17] F. Attneave, M. B., and D. O. Hebb, "The organization of behavior; a neuropsychological theory," *Amer. J. Phys. Med. Rehabil.*, vol. 30, no. 1, pp. 74–76, 2013.
[18] Y. Chauvin and D. E. Rumelhart, *Backpropagation: Theory, Architectures, and Applications*. Brighton, U.K.: Psychology Press, 1995.
[19] J. N. J. Reynolds, B. I. Hyland, and J. R. Wickens, "A cellular mechanism of reward-related learning," *Nature*, vol. 413, no. 6851, pp. 67–70, 2001.
[20] T. V. P. Bliss and G. L. Collingridge, "A synaptic model of memory: Long-term potentiation in the hippocampus," *Nature*, vol. 361, pp. 31–39, Jan. 1993.
[21] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning* (Springer Series in Statistics). Berlin, Germany: Springer, 2009.
[22] M. Olave, V. Rajkovic, and M. Bohanec, "An application for admission in public school systems," *Expert Systems in Public Administration*. Amsterdam, The Netherlands: Elsevier, 1989.
[23] B. Zupan, M. Bohanec, I. Bratko, and J. Demsăr, "Machine learning by function decomposition," in *Proc. 14th Int. Conf. Mach. Learn. (ICML)*, 1997, pp. 1–9.
[24] H.-R. Guo and Z.-M. Li, "A method of improving generalization ability for neural network based on genetic algorithm," in *Proc. IEEE Int. Conf. Intell. Comput. Intell. Syst.*, Oct. 2010, pp. 742–745.
[25] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning internal representations by error propagation," in *Neurocomputing: Foundations of Research*. Cambridge, MA, USA: MIT Press, 1988.
[26] T. Segaran, *Programming Collective Intelligence: Building Smart Web 2.0 Applications*. Sebastopol, CA, USA: O'Reilly Media, 2007.
[27] O. A. Jadaan, L. Rajamani, and C. R. Rao, "Improved selection operator for GA," *J. Theor. Appl. Inf. Technol.*, vol. 4, no. 4, pp. 269–277, 2008.
[28] R. Kohavi, "Scaling up the accuracy of Naive-Bayes classifiers: A decision-tree hybrid," in *Proc. 2nd Int. Conf. Knowl. Discovery Data Mining*, 1996, pp. 1–6.
[29] A. O. Aseeri, Y. Zhuang, and M. S. Alkatheiri, "A machine learning-based security vulnerability study on XOR PUFs for resource-constraint Internet of Things," in *Proc. IEEE Int. Congr. Internet Things*, Jul. 2018, pp. 49–56.
[30] J. Liang, E. Meyerson, B. Hodjat, D. Fink, K. Mutch, and R. Miikkulainen, "Evolutionary neural AutoML for deep learning," in *Proc. Genetic Evol. Comput. Conf.*, 2019, pp. 1–9.
[31] M. Feurer, A. Klein, K. Eggensperger, J. Springenberg, M. Blum, and F. Hutter, "Efficient and robust automated machine learning," in *Proc. Conf. Workshop Neural Inf. Process. Syst.*, 2015, pp. 2962–2970.

ZENGHAO CHAI is currently pursuing the degree with the School of Computer Science and Technology, Beijing Institute of Technology.



XU YANG received the Ph.D. degree in microelectronics from Tsinghua University, Beijing, China, in 2009. From October 2009 to February 2012, he was a Postdoctoral Researcher with the Department of Computer Science, Tsinghua University. Since March 2012, he has been with the Beijing Institute of Technology, Beijing. His research interests include intelligent transportation systems, artificial intelligence, and data science.

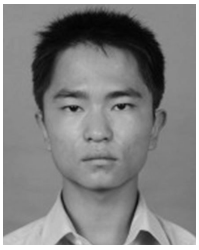
ZHILIN LIU is currently pursuing the degree with the School of Computer Science and Technology, Beijing Institute of Technology.



YUNLIN LEI is currently pursuing the degree with the School of Computer Science and Technology, Beijing Institute of Technology.

MENGYAO JI is currently pursuing the degree with the School of Computer Science and Technology, Beijing Institute of Technology.

WENHAO ZHENG is currently pursuing the degree with the School of Computer Science and Technology, Beijing Institute of Technology.



JINFENG ZHAO is currently pursuing the degree with the School of Computer Science and Technology, Beijing Institute of Technology.