SGNN: Synaptic-pruning and Genetic Neural Network

Li Tianyi

LiTianyiABC@outlook.com

Abstract

In the context of the development of artificial intelligence today, neural networks have become a key tool for solving complex problems. However, existing neural network architectures still face many challenges, including computational complexity, overfitting, and resource consumption. To address these issues, this paper proposes a novel neural network architecture - Synaptic-pruning and Genetic Neural Network (SGNN). This architecture combines biologically inspired synaptic-pruning mechanisms with genetic algorithms, dynamically adjusting the connections between neurons to optimize the network structure, thereby improving learning efficiency and computational performance. We designed SGN (Synaptic-pruning and Genetic Neuron), which not only has an adaptive activation threshold, but also can automatically enhance or inhibit synaptic strength based on performance during the learning process. The experimental results show that SGNN can effectively reduce overfitting and lower computational resource consumption while maintaining performance. This study provides new ideas for the design and optimization of neural networks, revealing the potential of combining synaptic-pruning with genetic algorithms, and opening up new directions for future neural network research.

Keywords: synaptic-pruning, genetic algorithm, neural network, biologically inspired

I. Introduction

A. Research background

With the rapid development of artificial intelligence, neural networks have been widely applied in fields such as image recognition, natural language processing, and medical diagnosis. The success of these applications has made neural networks the core of machine learning and deep learning research. However, despite the increasing capabilities of neural networks, challenges still exist in their architecture and training process, especially when dealing with increasingly complex tasks.

B. Limitations of existing neural network architectures

The current neural network architecture has limitations in multiple aspects. Firstly, many traditional models are prone to overfitting during the training process, resulting in insufficient generalization ability on new data. Secondly, as the number of network layers and parameters increases, the computational complexity significantly increases, leading to an increasing demand for computing resources. This not only increases training costs, but also limits its application in resource constrained environments. In addition, existing neuron designs often lack adaptability and cannot dynamically adjust their connection strength based on the characteristics of input data, which affects the efficiency and performance of the network.

C. Research purpose and significance

Given the aforementioned challenges, this paper aims to propose a novel neural network architecture – Synaptic-pruning and Genetic Neural Network (SGNN). The main highlight of SGNN lies in its introduction of genetic algorithms and synaptic-pruning and branch addition techniques. By dynamically adjusting the network structure, SGNN not only eliminates the fixed partitioning between the input layer, hidden layer, and output layer, but also achieves the flexibility that any neuron can serve as an input and output neuron. This design improves the adaptability of the network and allows it to quickly adjust and optimize the learning process in different tasks. The introduced genetic algorithm further enhances the adaptive ability of the network, enabling it to learn and evolve more effectively in complex environments**.**

II. Related work

A. Review of Traditional Neural Network Architecture

The research on traditional neural network architecture began in the 1980s, and the initial perceptron model laid the foundation for the subsequent Multilayer Perceptron (MLP). MLP utilizes backpropagation algorithm for training, enabling neural networks to solve nonlinear problems. However, traditional neural network architectures often face problems of overfitting and computational complexity. Despite the emergence of more advanced structures such as Convolutional Neural Networks (CNN) and Recurrent Neural Networks (RNN), the limitations of these models in handling dynamic tasks and adaptive capabilities remain significant. RNN can handle time series data, but it is prone to gradient vanishing and high computational overhead in long sequence training. Therefore, researchers have begun to explore more flexible and efficient network architectures to cope with complex application scenarios.

B. Research on Biological Neuron Model

The study of biological neuron models provides important insights for the development of neural networks. Biological neurons transmit information through electrical and chemical signals, with highly nonlinear and complex connectivity. In recent years, researchers have gradually incorporated the characteristics of biological neurons into computational models, such as Spiking Neural Networks (SNNs) and adaptive neural networks. These models demonstrate their potential in information processing and learning abilities by mimicking the working mechanisms of biological neurons. The theory of synaptic plasticity provides a theoretical basis for weight adjustment in networks and promotes in-depth research on adaptive learning mechanisms. These biologically inspired characteristics provide important references for the design of SGNN, enabling it to better simulate the processes of synaptic enhancement and inhibition.

C. The application of genetic algorithm in neural networks

Genetic algorithm, as an optimization technique based on natural selection and genetics principles, has been widely used in the design and training of neural networks. By simulating the process of biological evolution, genetic algorithms can effectively search for the optimal network structure and parameter configuration. Related studies have shown that genetic algorithms can provide good solutions in hyperparameter optimization, network architecture search, and feature selection. For example, genetic algorithms can select, crossover, and mutate the connections of neurons based on fitness evaluation, thereby enhancing the learning and generalization abilities of the network.

D. Synaptic-pruning and Genetic Neural Network (SGNN)

1. **Dynamic synaptic connections**: SGNN allows for dynamic adjustment of connections between neurons during training. By using the pruneSynapticConnections method and addSynapticConnections method in the code, the network can prune unnecessary synaptic connections in real-time based on the performance of neurons and add new connections as needed. This dynamism enables SGNN to more effectively respond to different task requirements.
2. **Adaptive learning mechanism**: SGNN calculates the error between the output of each neuron and the target output, and adjusts the synaptic strength based on this error. The updateSynapticStrengths method in the code demonstrates how to enhance or inhibit synaptic strength based on successful and failed experiences. This mechanism mimics the learning process of biological neurons, enabling SGNN to self optimize when facing complex environments.
3. **Genetic Algorithm Optimization**: SGNN integrates genetic algorithms to explore parameter space through crossover and mutation operations. The crossover and mutate methods in the code provide a mechanism for randomly adjusting the threshold and synaptic strength of neurons, thereby enhancing the adaptability and robustness of the network.
4. **Effect evaluation and feedback**: SGNN uses forward and backward propagation to calculate output and error, ensuring that the network can continuously adjust its own parameters based on real-time feedback. The calculation of the loss function is achieved through the computeLoss method, ensuring that the learning process of the network always converges towards the optimal solution.

Through the above design, SGNN not only improves the adaptive ability of neural networks, but also significantly enhances their performance in complex tasks. This diversified design provides new ideas and directions for the further development of neural networks.

III. SGNN architecture design

A. Overview of Overall Architecture

SGNN (Synaptic-pruning and Genetic Neural Network) is an innovative neural network architecture designed to simulate the characteristics of biological nervous systems for adaptive learning and optimization. This architecture combines synaptic-pruning and genetic algorithm mechanisms, which can effectively handle complex tasks and quickly adapt to environmental changes. The design goal of SGNN is to improve the learning efficiency of the network by dynamically adjusting the connection strength and threshold of neurons, and optimize the network structure through genetic algorithm.

1. Main components

- **Neuron**: The basic computing unit responsible for receiving, processing, and outputting signals.
- **Synaptic connections**: Information transmission channels between neurons, whose strength determines the effectiveness of signal transmission.
- **Learning mechanism**: Through the enhancement and inhibition of synapses, the network

achieves learning and memory.

2. Design objectives

- Implement dynamic adaptive learning capability to cope with changing input data.
- Reducing redundant connections and improving computational efficiency through synaptic-pruning.
- Optimizing network structure and parameters through genetic algorithm to enhance network performance.

B. Neuron architecture design

1. Neuron model parameters

Each neuron has the following attributes:

- **Output value (y)**: represents the activation state of the neuron.
- **Threshold (θ)**: The critical value that determines the activation of neurons.
- **Synaptic connection strength (w_{ij})**: The strength of connections with other neurons that affects the transmission of information.
- **Enhance Count (E)**: Record the number of successful activations.
- **Suppression Count (I)**: Record the number of failed activations.

2. Activation function

SGNN uses an activation function called **AaS (Activation and Suppression)**, defined as:

$$y = f(x) = \frac{1 - e^{-x+\theta}}{1 + e^{-x+\theta}}$$

This function can effectively suppress neurons after activation, thereby enhancing the dynamic response capability of the network.

3. Connection mechanism

- **Synaptic enhancement and inhibition:**
  - When the output of a neuron is successful, enhance its synaptic connections:
    $$w_{ij} \leftarrow w_{ij} + \Delta w(\text{succeed})$$
  - When the output fails, suppress its synaptic connections:
    $$w_{ij} \leftarrow w_{ij} - \Delta w(\text{failed})$$
- **Synaptic-pruning and branching:**
  - **Synaptic-pruning**: If the synaptic strength is below the preset threshold, perform pruning
    $$\text{if } |w_{ij}| < \text{PRUNE\_THRESHOLD} \Rightarrow \text{remove } w_{ij}$$
  - **Synaptic-branching**: Add new synapses based on randomly generated conditions
    $$\text{if rand}() < P \text{ and } |\text{synapticStrengths}| < \frac{N}{2} \Rightarrow \text{add } w_{ij}$$

C. Network training methods

1. Training a single neural network

The training process of SGNN begins with iterative training of a single neural network:

- **Initialization**: At the beginning of training, randomly initialize the threshold (θ) and synaptic connection strength (w_{ij})for each neuron.
- **Forward propagation:**
  - **Weighted input calculation**: The weighted input of each neuron is the sum of the

product of the output from other connected neurons and the synaptic strength. For the number (i) neuron, its weighted input (x_i) is represented as:

$$x_i = \sum_j w_{ij} \cdot y_j$$

- **Application activation function**: The output of neurons is calculated through the activation function

$$y_i = f(x_i) = \frac{1 - e^{-x_i + \theta_i}}{1 + e^{-x_i + \theta_i}}$$

- **Loss calculation**: Evaluate the performance of the network using Mean Absolute Error (MAE):

$$loss = \frac{1}{N} \sum_{k=1}^{N} | y_k - \hat{y}_k |$$

Among them, (y_k) is the actual output, and ($\hat{y}_k$) is the target output.

- **Backpropagation:**
  - **Calculate output error**: For each output neuron, calculate the absolute error between its output and the target output:

$$\delta_k = y_k - \hat{y}_k$$

  - **Update synaptic weights**:

$$w_{ij} \leftarrow w_{ij} - \alpha \cdot \delta_k \cdot y_j$$

Among them, (α) is the learning rate, and (δ_k) is the corresponding output error.

- **Update global threshold**:

$$\theta_i \leftarrow \theta_i - \beta \cdot \text{average}(\delta)$$

Among them, (β) is the global threshold adjustment rate, and (average(δ)) is the average of all output errors.

2. Achieve the maximum number of iterations

When the set maximum number of iterations is reached, the model stops training and records the current optimal parameters and state. These optimal parameters will be used to generate a new generation of neural networks.

3. Breeding new neural networks

After training, the trained neural network is used as the parent neural network, from which multiple sub neural networks are propagated:

- **Selection**: Choose the parent neural network with the highest fitness.
- **Crossover**: Generating new sub neural networks through gene crossover:

$$w_{ij}^{\text{child}} = \begin{cases} w_{ij}^{\text{parent1}} \text{ with probability } 0.5 \\ w_{ij}^{\text{parent2}} \text{ with probability } 0.5 \end{cases}$$

4. Iterative optimization

Perform the following steps on the newly generated sub neural network to optimize performance:

- **Mutation**: Randomly adjusting the synaptic connections of sub neural networks to increase variability.

$$w_{ij} \leftarrow w_{ij} + \text{random adjustment (According to the Rate of Variation)}$$

- **Training and Evaluation**: Train all sub neural networks, evaluate their performance, and select the sub neural network with the highest fitness as the parent neural network for the next generation.

5. Termination and Model Saving

The entire training process terminates when the fitness reaches the preset standard, and the current best model parameters are saved to ensure that the optimal state can be quickly loaded for future use.

Through this forward and backward propagation mechanism, SGNN can continuously optimize its network structure, improve learning efficiency and adaptability, thereby exhibiting stronger performance in complex tasks.

IV. Experimental design and evaluation

1. Small scale concept validation

In the initial development stage of the SGNN (Synaptic-pruning and Genetic Neural Network) model, multiple experiments were conducted to verify its design concept and basic functions, in order to confirm the effectiveness and feasibility of the model. The experiments for concept validation mainly focus on training and testing models using small datasets to quickly validate their basic abilities.

1.1 Dataset selection

We use a small dataset that contains a set of sample input data and its corresponding sample output data. The characteristics of this dataset are as follows:

- **Input data**: A set of feature vectors used to simulate the input of a neural network.
- **Output data**: The target output corresponding to the input data is a continuous numerical value used for fitting.

The selection of this dataset allows SGNN to be trained in a shorter amount of time and quickly evaluate its performance on data fitting tasks.

1.2 Experimental steps

1. Data preparation and training platform: Prepare input and output data. The training platform uses **Intel Core i7-10510U CPU**.

2. Model initialization: Randomly set the threshold and synaptic connection strength of neurons to establish an SGNN model.

3. Training process: Perform multiple rounds of training on SGNN and record the loss value and model output for each round. Optimize training effectiveness by adjusting hyperparameters such as learning rate, synaptic enhancement, and inhibition thresholds.

4. Performance metrics: Use metrics such as loss during regular training and loss during genetic algorithm training to evaluate the performance of the model on the training and testing sets.

1.3 Result analysis

- **Output result**: After the training is completed, record the errors of the model on the training and testing sets, and compare the differences between the actual output and the sample output.

  The regular training consists of 1000 steps, while the genetic algorithm is trained for 10 generations

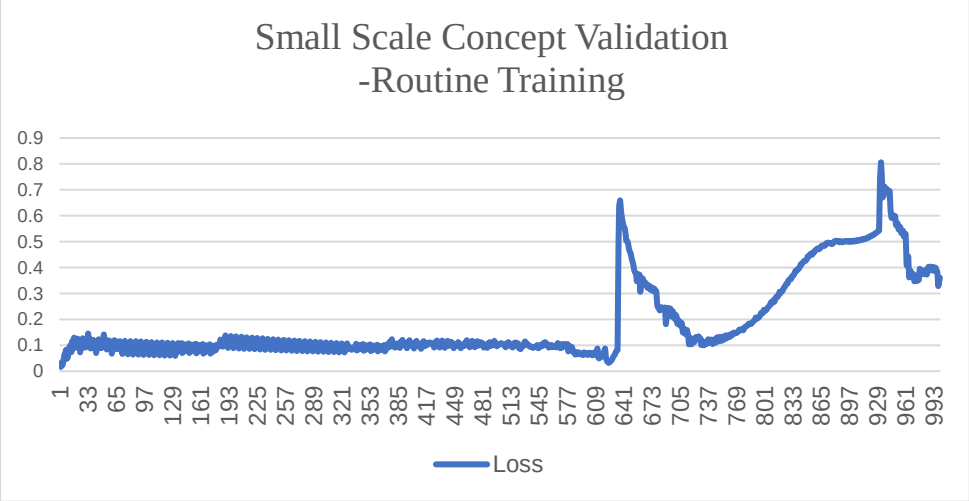
Model actual output: Neuron 18 Output: -0.109741 Neuron 19 Output: 0.730713

Model example output: Neuron 18 Output: -0.1 Neuron 19 Output: 0.7

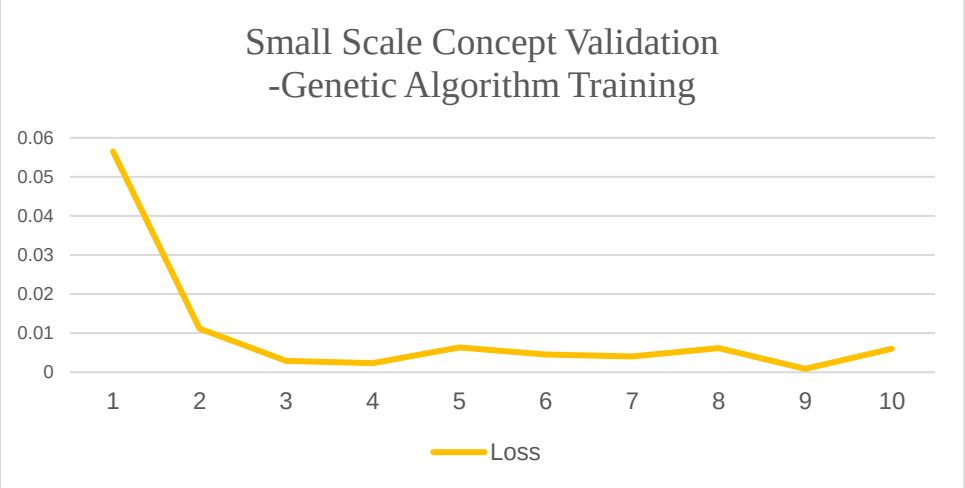
From the actual output of the trained model above, it can be seen that the actual output of the model is very close to the sample output, which proves that the concept of SGNN model is correct.

- **Learning curve**: The learning curve of SGNN when fitting examples.

The following is the change in loss during regular training:



The following is the change in loss during genetic algorithm training:



- **Model training time**: 0.9014s

2. Large scale fitting test

In order to comprehensively evaluate the performance of SGNN, tests were conducted, and the analysis of the test results included training time, parameter size, training effectiveness, etc.

2.1 Experimental steps

1. Data preparation and training platform: Prepare input and output data. The training platform uses **Intel Core i7-10510U CPU**.

2. Model initialization: Randomly set the threshold and synaptic connection strength of neurons to establish an SGNN model.

3. Training process: Perform multiple rounds of training on SGNN and record the loss value

and model output for each round. Optimize training effectiveness by adjusting hyperparameters such as learning rate, synaptic enhancement, and inhibition thresholds.

4. Performance metrics: Use metrics such as loss during regular training and loss during genetic algorithm training to evaluate the performance of the model on the training and testing sets.

2.2 Performance indicators

- **Training time**: The time used for this test is 43.17 seconds
- **Model complexity**: The model has 100 neurons and a total parameter size of 64KB
- **Convergence speed**: In regular training, the loss variation of the model is unstable, while in genetic algorithm training, the loss of the model does not exceed 0.05.
- **Model output**: After training, compare the actual output and sample output of the model to evaluate its performance.

  Model actual output: Neuron 98 Output: 0.460085 Neuron 99 Output: -0.259907

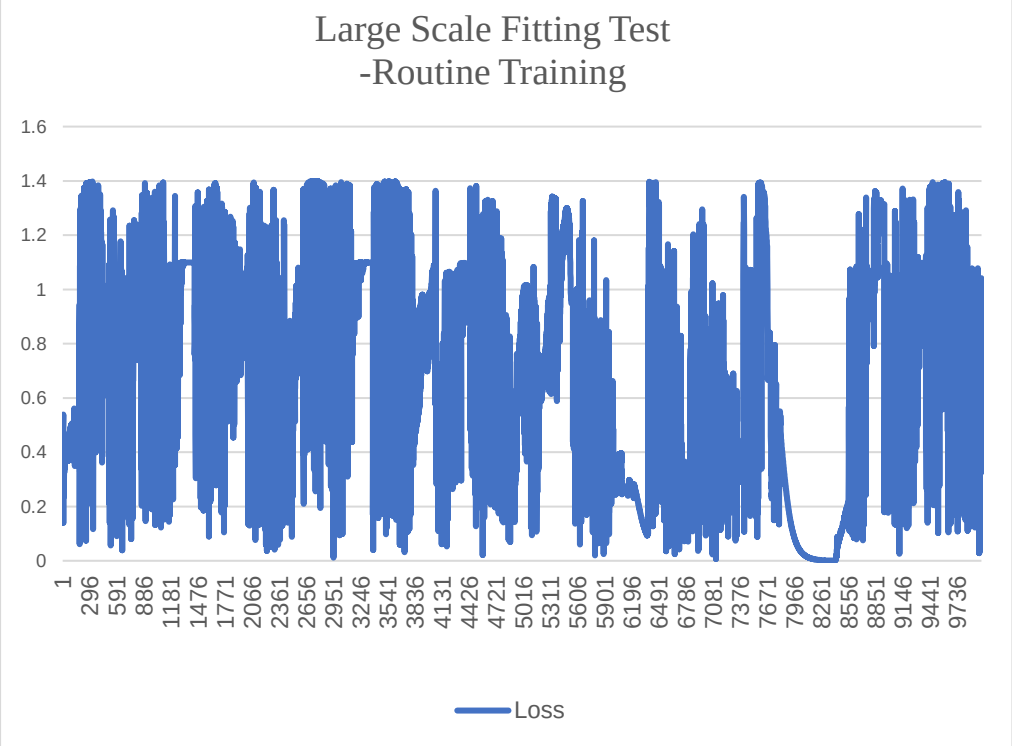
  Model example output: Neuron 98 Output: 0.5 Neuron 99 Output: -0.3

  From the actual output of the trained model above, it can be seen that the actual output of the model is very close to the sample output, which indicates that SGNN can maintain good performance on a large scale with a large number of parameters.

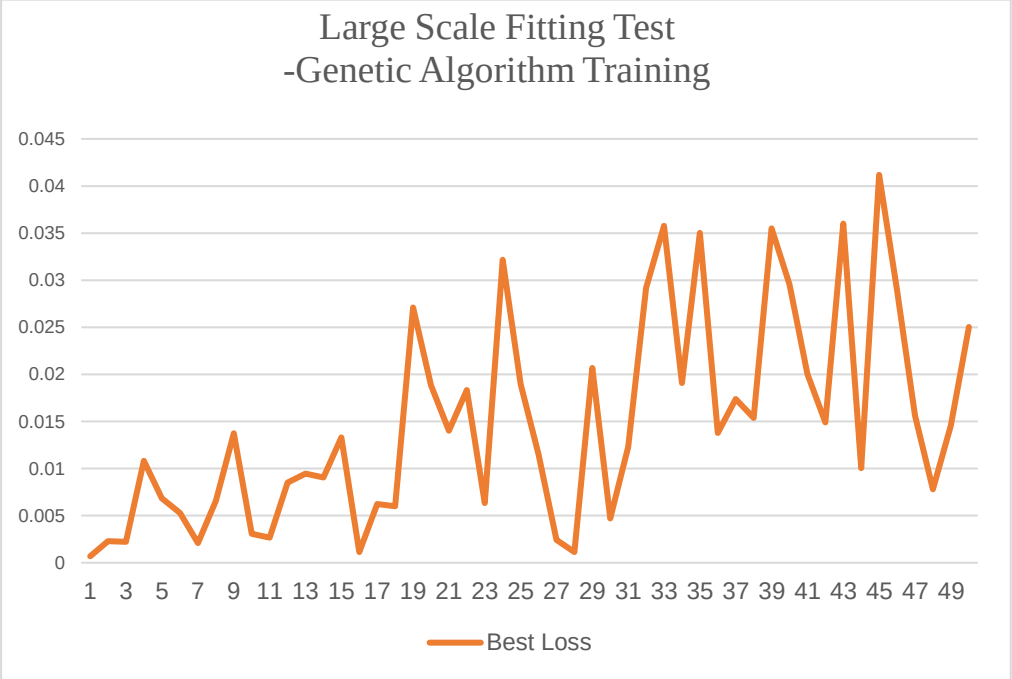
2.3 Result Analysis

- **Visualization display**: The following shows the curve of error variation with iteration times during SGNN training.

  The loss changes during regular training are as follows:



The optimal loss variation in genetic algorithm training is as follows:



- **Comparative analysis**: The loss variation of SGNN is not stable during the training process, but in subsequent genetic algorithm training, the model's loss is relatively low, which reflects the excellent performance of SGNN.

3. Multi-sample test

3.1 Test data and training steps

- **Test data**: This test used 4 sets of data, each including a set of sample inputs and a set of sample outputs.
- **Model training steps:**

  1. Data preparation and training platform: Prepare input and output data. The training platform uses **Intel Core i7-10510U CPU**.

  2. Model initialization: Randomly set the threshold and synaptic connection strength of neurons to establish an SGNN model.

  3. Training process: Perform multiple rounds of training on SGNN and record the loss value and model output for each round. Optimize training effectiveness by adjusting hyperparameters such as learning rate, synaptic enhancement, and inhibition thresholds.

  4. Performance metrics: Use metrics such as loss during regular training and loss during genetic algorithm training to evaluate the performance of the model on the training and testing sets.
- **Training steps**: Conventional training takes 200 steps, while genetic algorithm training takes 1000 generations.

3.2 Performance indicators

- **Training time**: The time used for this test is 10.01 seconds
- **Model complexity**: The model has a total of 20 neurons and a total parameter size of 3KB
- **Convergence speed**: In regular training, the model experiences unstable training losses in the early stages and stable losses of around 0.6 in the later stages. In genetic algorithm training, the overall training loss of the model shows a decreasing trend and eventually

approaches zero.

- **Model output**: After training, compare the actual output and sample output of the model to evaluate its performance.

  Actual output of the model:

  Group 1:

  Predicted Outputs: -0.105969 0.724131

  Group 2:

  Predicted Outputs: 0.558556 -0.31867

  Group 3:

  Predicted Outputs: 0.287975 0.726394

  Group 4:

  Predicted Outputs: 0.670793 -0.244593

  Model example output:

  Group 1:

  Sample Outputs: -0.1 0.7

  Group 2:

  Sample Outputs: 0.5 -0.3

  Group 3:

  Sample Outputs: 0.3 0.8

  Group 4:

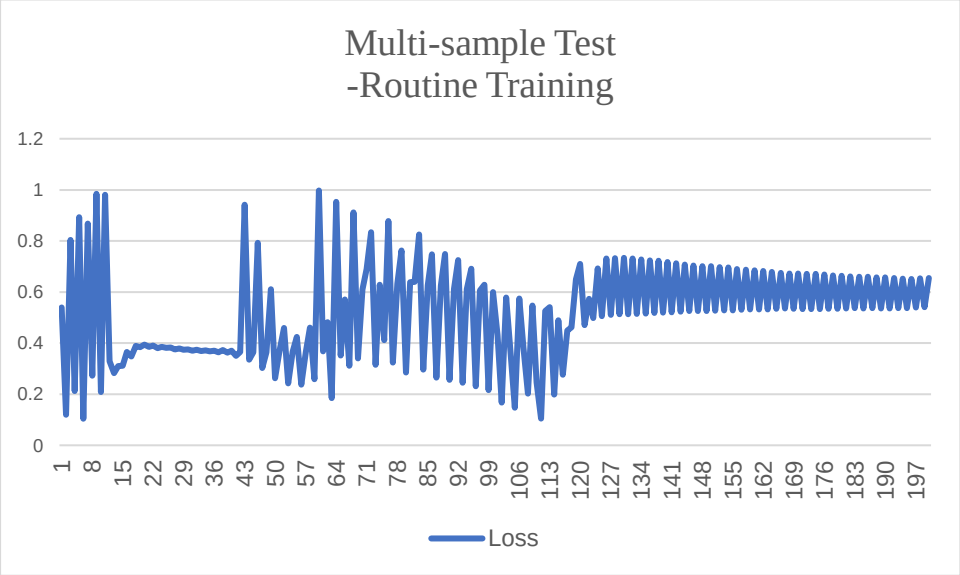
  Sample Outputs: 0.7 -0.1

  From the actual output of the trained model above, it can be seen that the actual output of the model is very close to the sample output, which indicates that SGNN can also maintain good performance in multi sample fitting.

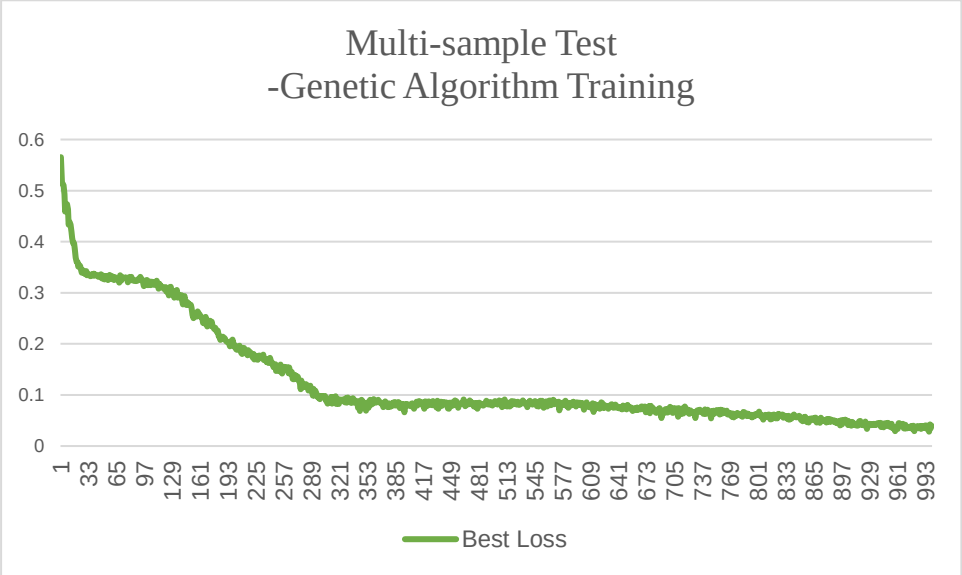
3.3 Result Analysis

- **Visualization display**: The following shows the curve of error variation with iteration times during SGNN training.

  The loss changes during regular training are as follows:



The optimal loss variation in genetic algorithm training is as follows:



- **Comparative analysis**: SGNN can complete the fitting of multiple samples in a relatively short time in multi sample testing, which reflects the flexibility of SGNN.

V. Discussion

A. Advantages and Innovations of SGNN

1. Dynamic synaptic connections

The core innovation of SGNN lies in its dynamic synaptic connection mechanism. By combining synaptic pruning and branching techniques, SGNN can adjust the network structure in real-time according to the requirements of the current task. This mechanism makes the connections between neurons no longer static, but can constantly change with feedback during the learning process. For example, when a neuron performs poorly, the synapses connected to it can be effectively pruned to reduce the transmission of redundant information; On the contrary, if a connection performs well in a task, its strength can be increased. This dynamic adjustment not only optimizes the network structure, but also significantly improves its adaptability, enabling SGNN to quickly respond to changing input data in different application scenarios.

2. The adaptive learning mechanism

SGNN further enhances its learning ability through adaptive learning mechanisms. In this mechanism, the enhancement and inhibition of synapses are automatically regulated based on the performance of neurons during the learning process. When neurons are successfully activated, the strength of their connections will be enhanced; When activation fails, it will be suppressed. This design enables SGNN to embody a learning process similar to that of biological neurons, continuously optimizing its connection patterns through empirical feedback. This adaptive mechanism effectively improves learning efficiency, enabling SGNN to exhibit higher accuracy and robustness when facing complex data.

3. Genetic Algorithm Optimization

The introduction of genetic algorithm optimization is another major innovation of SGNN, which endows the model with the ability to self evolve. By simulating the process of natural selection, SGNN can continuously optimize its network structure and parameter configuration in multiple generations of training. The crossover and mutation operations of genetic algorithms

enable SGNN to explore a wider parameter space and find better solutions. This self evolving ability enables SGNN to not only adapt to new task requirements when dealing with complex environments, but also effectively improve the model's generalization ability.

B. Limitations and challenges that exist

1. Computational complexity

Although SGNN has shown significant advantages in network structure optimization, its training process still faces high computational complexity. When processing large-scale datasets, SGNN requires more computing resources, including longer training time and higher memory consumption. This may result in low efficiency for resource constrained environments or real-time applications. Therefore, how to optimize the computational performance of SGNN and reduce its dependence on hardware resources is an important direction for future research.

2. Parameter selection

The performance of SGNN largely depends on the hyperparameter settings in genetic algorithms and synaptic-pruning mechanisms. However, there is currently a lack of systematic guidance for the selection of these parameters, and setting them reasonably remains a challenge. For example, the threshold for synaptic-pruning, the crossover rate and mutation rate of genetic algorithms are key factors that affect model performance. Future research needs to explore effective hyperparameter optimization strategies to ensure that SGNN can achieve optimal performance in different applications.

3. Model interpretability

Due to the dynamic changes in the network structure of SGNN, the interpretability of its learning process is relatively low. The current interpretability research mainly focuses on static models, and for SGNN with dynamic and adaptive characteristics, existing methods may be difficult to effectively apply. In order to improve the interpretability of the model, more research is needed in the future to explore how to reveal the internal working mechanism of SGNN through visualization techniques and network structure analysis, making its decision-making process more transparent.

C. Future improvement direction

1. Efficient training strategy

To improve the training efficiency of SGNN on large-scale datasets, future research can focus on developing more efficient training strategies. For example, distributed training and parallel computing can significantly accelerate the training process. In addition, utilizing modern hardware such as GPUs and TPUs for accelerated training may enable SGNN to handle larger scale tasks while maintaining high training performance.

2. Adaptive hyperparameter adjustment

The introduction of adaptive hyperparameter adjustment mechanism will enable SGNN to automatically optimize parameter configuration when facing different tasks and environments. By monitoring the performance of the model in real-time and dynamically adjusting hyperparameters, SGNN can continuously adapt to new conditions during the training process, improving its generalization ability and robustness. The implementation of this mechanism will have a profound impact on the application scope of SGNN.

3. Improve interpretability

Improving interpretability and enhancing the interpretability of SGNN is one of the key directions for future research. Through visualization and network structure analysis, researchers can better understand the internal mechanisms of SGNN, such as exploring which synaptic connections and neuronal activity have a significant impact on final decisions. In addition, developing interpretable tools and methods for SGNN will help users understand their decision-making process, thereby enhancing trust in the model, especially in high-risk areas such as healthcare and finance.

Through the above discussion, we can see the potential and value of SGNN in the field of neural networks, while also realizing the challenges and room for improvement it faces in practical applications. Future research will focus on addressing these issues to further promote the practical application and development of SGNN.

VI. Conclusion

This article proposes a novel neural network architecture – Synaptic-pruning and Genetic Neural Network (SGNN), which combines synaptic-pruning and branch addition mechanisms with genetic algorithm optimization, aiming to improve the learning efficiency and computational performance of neural networks. The experimental results show that SGNN can effectively reduce the complexity of the network, improve computational efficiency, and demonstrate good generalization ability in different tasks while maintaining performance. Future research directions include developing more efficient training strategies, introducing adaptive hyperparameter adjustment mechanisms, and improving model interpretability.

VII. Appendix

A. Model hyperparameter settings

1. Small scale concept validation

Routine Training Hyperparameters	
LEARNING_RATE	**0.01**
PRUNE_THRESHOLD	**0.1**
ADD_THRESHOLD	**0.5**
NUM_NEURONS	**20**
MAX_ITERATIONS	**1000**
GLOBAL_THRESHOLD_ADJUSTMENT_RATE	**0.01**
LONG_TERM_STRENGTHEN	**0.05**
LONG_TERM_INHIBIT	**0.05**
THRESHOLD_FOR_ENHANCE	**3**
THRESHOLD_FOR_INHIBIT	**3**
Genetic Algorithm Training Hyperparameters	
POPULATION_SIZE	**100**
MUTATION_RATE	**0.1**
NUM_GENERATIONS	**50**

2. Large scale fitting test

Routine Training Hyperparameters	
LEARNING_RATE	0.001
PRUNE_THRESHOLD	0.1
ADD_THRESHOLD	0.5
NUM_NEURONS	100
MAX_ITERATIONS	10000
GLOBAL_THRESHOLD_ADJUSTMENT_RATE	0.01
LONG_TERM_STRENGTHEN	0.05
LONG_TERM_INHIBIT	0.05
THRESHOLD_FOR_ENHANCE	3
THRESHOLD_FOR_INHIBIT	3
Genetic Algorithm Training Hyperparameters	
POPULATION_SIZE	100
MUTATION_RATE	0.01
NUM_GENERATIONS	50

3. Multi-sample test

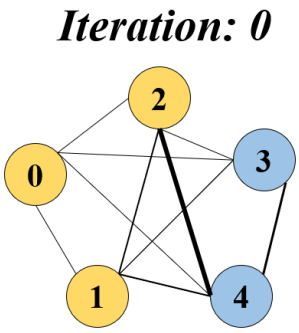
Routine Training Hyperparameters	
LEARNING_RATE	0.001
PRUNE_THRESHOLD	0.1
ADD_THRESHOLD	0.5
NUM_NEURONS	20
MAX_ITERATIONS	200
GLOBAL_THRESHOLD_ADJUSTMENT_RATE	0.01
LONG_TERM_STRENGTHEN	0.05
LONG_TERM_INHIBIT	0.05
THRESHOLD_FOR_ENHANCE	3
THRESHOLD_FOR_INHIBIT	3
Genetic Algorithm Training Hyperparameters	
POPULATION_SIZE	100
MUTATION_RATE	0.01
NUM_GENERATIONS	1000

B. Example of Network Structure Evolution

Under the influence of synaptic-pruning and branching mechanisms, the network structure of SGNN continuously evolves during the training process. Taking an SGNN with 5 neurons as an example, demonstrate the evolution process of the network structure.

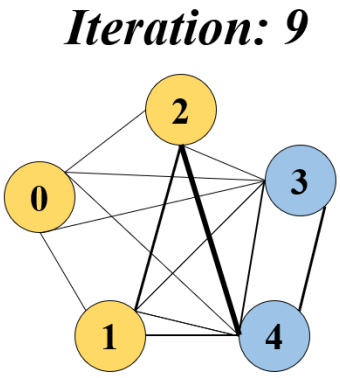
1. Initial network structure

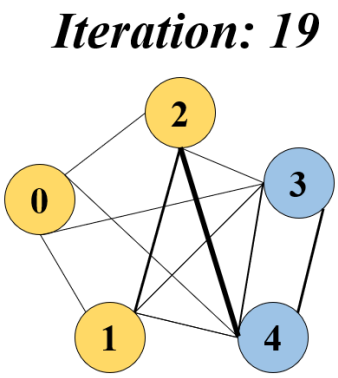
In the initial state, the neurons of SGNN are randomly connected to form a relatively complex network structure. The following is the initial network structure:



2. Evolution process of network structure

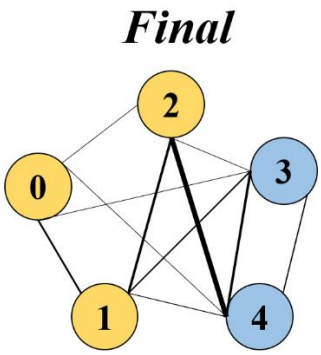
During the training process, the synaptic-pruning mechanism continuously removes synapses with weaker connection strength, while the branching mechanism adds new synaptic connections. The network structure after several rounds of training is as follows:





3. Final network structure

The final network structure after multiple iterations and genetic algorithm optimization is as follows:



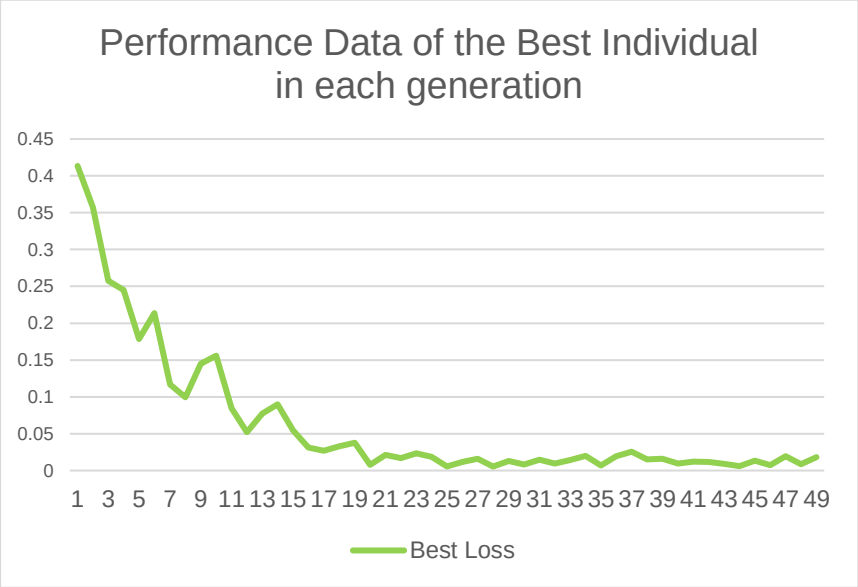
Through this evolutionary process, the network structure of SGNN gradually converges to a more concise and effective state, achieving the best fit for the target task. Meanwhile, the crossover and mutation operations of genetic algorithms further enhance the diversity and adaptability of the network.

Through the design and experimental analysis of this article, SGNN has demonstrated significant advantages in performance and structural optimization in different tasks, providing new ideas and directions for the design and optimization of future neural networks.

C. Detailed data on the optimization process of genetic algorithm

In the optimization process of genetic algorithm, we continuously optimize the network structure and parameter configuration of SGNN through multiple iterations and selection, crossover and mutation operations. The following is the performance data of the best individual in each generation. (Taking a 20 neuron network as an example here)

1. Performance data of the best individual in each generation



From this data, it can be seen that during the optimization process of genetic algorithm, the error of SGNN gradually decreases, indicating that the performance of the model is constantly improving.

2. The impact of different hyperparameter settings

In order to evaluate the impact of different hyperparameter settings on the performance of SGNN in genetic algorithms, we designed a series of experiments to compare the performance of the model under different population sizes and mutation rate settings.

The following explores the impact of population size on performance, maintaining a mutation rate of 0.01 and changing population size

When the population size is 10, the optimal individual training error for each generation is shown in the following table:

Generation	Best Loss
0	0.76365
1	0.415553
2	0.668207
3	0.354502
4	0.706213
5	0.315129
6	0.730687
7	0.299143
8	0.744576
9	0.297102

When the population size is 20, the optimal individual training error for each generation is shown in the following table:

Generation	Best Loss
0	0.144946
1	0.124349
2	0.117603
3	0.113168
4	0.109033
5	0.105308
6	0.0937293
7	0.0898182
8	0.0972035
9	0.0726572

When the population size is 50, the optimal individual training error for each generation is shown in the following table:

Generation	Best Loss
0	0.0561872
1	0.0173442
2	0.0149225
3	0.0305253
4	0.0124321
5	0.0127186
6	0.0160219
7	0.0121966
8	0.00815077
9	0.00617266

When the population size is 100, the optimal individual training error for each generation is shown in the following table:

Generation	Best Loss
0	0.0307105
1	0.0153517
2	0.0172473
3	0.0134103
4	0.00229408
5	0.00540447
6	0.00883831
7	0.00460834
8	0.00178654
9	0.00214223

From the above data, it can be found that the larger the population size, the smaller the training error of the optimal individual in each generation during genetic algorithm training, and the better the performance of the trained model.

The following explores the impact of mutation rate on performance, keeping the population size at 50 and changing the mutation rate

When the mutation rate is 0.01, the optimal individual training error for each generation is

shown in the following table:

Generation	Best Loss
0	0.0561872
1	0.0173442
2	0.0149225
3	0.0305253
4	0.0124321
5	0.0127186
6	0.0160219
7	0.0121966
8	0.00815077
9	0.00617266

When the mutation rate is 0.05, the optimal individual training error for each generation is shown in the following table:

Generation	Best Loss
0	0.381955
1	0.121145
2	0.347678
3	0.0795475
4	0.304961
5	0.0489211
6	0.297496
7	0.0125185
8	0.308884
9	0.00855552

When the mutation rate is 0.1, the optimal individual training error for each generation is shown in the following table:

Generation	Best Loss
0	0.919423
1	0.783771
2	0.0116596
3	0.817741
4	0.788749
5	0.0149593
6	0.72605
7	0.717706
8	0.0401251
9	0.497469

When the mutation rate is 0.5, the optimal individual training error for each generation is shown in the following table:

Generation	Best Loss
0	0.211619
1	0.384163

2	**0.963564**
3	**0.798797**
4	**0.307409**
5	**0.253193**
6	**0.0423309**
7	**0.336931**
8	**0.405121**
9	**0.246261**

From the above data, it can be found that the smaller the mutation rate, the smaller the training error of the optimal individual in each generation during genetic algorithm training, and the better the performance of the trained model.

D. Code Usage Guide

For the convenience of readers reproducing the experimental results in this article, the following provides a code usage guide for SGNN.

1. Environment configuration

Firstly, ensure that the C++compiler is installed in the system and has support for the C++standard library.

2. Compile and Run

Save the provided code to a. cpp file (using sgnn.cpp as an example), and then compile and run it using the following command:

g++ sgnn.cpp -o sgnn -std=c++11

./sgnn

Note: The provided code must use the C++11 standard as a minimum.

3. Save and Read Results

The code provides saveParameters and loadParameters functions for saving and reading model parameters. The results will be saved in the file parameters.txt.

E. Model Connection Structure

1. Small scale concept validation

The following is the internal connectivity of the model after small-scale concept validation training.

Neuron 0 - Last Output: -0.526991, Threshold: 0.424345, Connections: (Neuron 1, Strength: 0.289163) (Neuron 5, Strength: -0.484255) (Neuron 11, Strength: -0.481332) (Neuron 13, Strength: 0.481491) (Neuron 19, Strength: 0.764183) (Neuron 5, Strength: -0.626615) (Neuron 6, Strength: -0.657305) (Neuron 7, Strength: 0.279919) (Neuron 12, Strength: 0.222257) (Neuron 15, Strength: 0.749416)

Neuron 1 - Last Output: 0.329221, Threshold: 0.119181, Connections: (Neuron 4, Strength: -0.169811) (Neuron 5, Strength: -0.356404) (Neuron 7, Strength: 0.826313) (Neuron 10, Strength: -0.424247) (Neuron 12, Strength: -0.255538) (Neuron 14, Strength: -0.289834) (Neuron 16, Strength: 0.368795) (Neuron 17, Strength: -0.475426) (Neuron 0, Strength: -0.846404) (Neuron 3, Strength: -0.936399)

Neuron 2 - Last Output: -0.580061, Threshold: 0.563363, Connections: (Neuron 5, Strength: 0.41081) (Neuron 6, Strength: -0.817847) (Neuron 9, Strength: -0.264934) (Neuron 10, Strength: 0.314054) (Neuron 3, Strength: -0.860836) (Neuron 4, Strength: 0.770602) (Neuron 6, Strength: 0.891449) (Neuron 7, Strength: 0.150365) (Neuron 4, Strength: 0.482974) (Neuron 6, Strength: 0.184625)

Neuron 3 - Last Output: -0.0309188, Threshold: 0.148168, Connections: (Neuron 8, Strength: -0.26719) (Neuron 9, Strength: 0.963805) (Neuron 10, Strength: -0.751216) (Neuron 11, Strength: 0.354366) (Neuron 13, Strength: 0.780068) (Neuron 14, Strength: 0.239707) (Neuron 16, Strength: 0.940172) (Neuron 17, Strength: -0.437346) (Neuron 18, Strength: -0.561889) (Neuron 0, Strength: 0.772528)
Neuron 4 - Last Output: -0.465036, Threshold: 0.586777, Connections: (Neuron 2, Strength: 0.573617) (Neuron 3, Strength: 0.823649) (Neuron 5, Strength: 0.273956) (Neuron 7, Strength: -0.729942) (Neuron 9, Strength: 0.885372) (Neuron 11, Strength: -0.0859435) (Neuron 12, Strength: 0.628834) (Neuron 13, Strength: -0.87788) (Neuron 14, Strength: -0.909735) (Neuron 15, Strength: -0.58005) (Neuron 17, Strength: -0.734672) (Neuron 19, Strength: -0.727744)
Neuron 5 - Last Output: -0.0811299, Threshold: 0.777459, Connections: (Neuron 6, Strength: 0.562712) (Neuron 7, Strength: 0.918656) (Neuron 8, Strength: 0.891598) (Neuron 9, Strength: -0.735676) (Neuron 11, Strength: -0.970974) (Neuron 12, Strength: -0.945982) (Neuron 14, Strength: 0.427351) (Neuron 15, Strength: -0.431895) (Neuron 16, Strength: 0.984927) (Neuron 19, Strength: 0.560317)
Neuron 6 - Last Output: 0.262297, Threshold: 0.64807, Connections: (Neuron 0, Strength: -0.552809) (Neuron 2, Strength: -0.934913) (Neuron 13, Strength: 0.932218) (Neuron 15, Strength: -0.580621) (Neuron 16, Strength: 0.0710262) (Neuron 17, Strength: 0.84842) (Neuron 1, Strength: -0.18547) (Neuron 7, Strength: 0.372713) (Neuron 5, Strength: -0.413126) (Neuron 13, Strength: 0.188131)
Neuron 7 - Last Output: -0.256522, Threshold: -0.140269, Connections: (Neuron 3, Strength: 0.496088) (Neuron 4, Strength: 0.378548) (Neuron 6, Strength: -0.843928) (Neuron 9, Strength: 0.2618) (Neuron 10, Strength: -0.409281) (Neuron 14, Strength: 0.323283) (Neuron 18, Strength: -0.658177) (Neuron 5, Strength: -0.645143) (Neuron 6, Strength: -0.880606) (Neuron 1, Strength: -0.919739)
Neuron 8 - Last Output: -0.494109, Threshold: 0.657693, Connections: (Neuron 1, Strength: 0.112949) (Neuron 2, Strength: -0.292831) (Neuron 5, Strength: -0.764986) (Neuron 9, Strength: 0.538868) (Neuron 10, Strength: -0.0917693) (Neuron 13, Strength: 0.630473) (Neuron 17, Strength: -0.930045) (Neuron 19, Strength: 0.674471) (Neuron 0, Strength: 0.841627) (Neuron 1, Strength: -0.436027)
Neuron 9 - Last Output: 0.00316745, Threshold: 0.615366, Connections: (Neuron 0, Strength: 0.994744) (Neuron 4, Strength: -0.107727) (Neuron 5, Strength: -0.87058) (Neuron 8, Strength: 0.38327) (Neuron 12, Strength: -0.848949) (Neuron 16, Strength: 0.350905) (Neuron 17, Strength: 0.965163) (Neuron 19, Strength: 0.546965) (Neuron 3, Strength: 0.582467) (Neuron 12, Strength: 0.353685)
Neuron 10 - Last Output: -0.646859, Threshold: 0.178943, Connections: (Neuron 0, Strength: 0.293631) (Neuron 2, Strength: 0.543739) (Neuron 3, Strength: -0.124216) (Neuron 4, Strength: -0.704187) (Neuron 6, Strength: -0.910736) (Neuron 7, Strength: -0.99857) (Neuron 13, Strength: 0.814115) (Neuron 15, Strength: 0.829606) (Neuron 16, Strength: 0.702658) (Neuron 17, Strength: -0.264931) (Neuron 18, Strength: 0.0549548) (Neuron 19, Strength: -0.506018)
Neuron 11 - Last Output: 0.360864, Threshold: 0.657357, Connections: (Neuron 1, Strength: 0.898743) (Neuron 2, Strength: -0.480871) (Neuron 8, Strength: 0.594158) (Neuron 9, Strength: 0.409274) (Neuron 13, Strength: -0.798004) (Neuron 15, Strength: -0.714719) (Neuron 16, Strength: 0.421207) (Neuron 17, Strength: -0.287622) (Neuron 1, Strength: -0.745613) (Neuron 2, Strength:

-0.920416)
Neuron 12 - Last Output: -0.639451, Threshold: 0.629362, Connections: (Neuron 3, Strength: -0.164913) (Neuron 5, Strength: 0.842952) (Neuron 6, Strength: -0.441395) (Neuron 10, Strength: 0.931922) (Neuron 11, Strength: -0.49512) (Neuron 15, Strength: -0.342561) (Neuron 16, Strength: -0.319505) (Neuron 18, Strength: -0.79567) (Neuron 0, Strength: 0.431633) (Neuron 3, Strength: -0.467284)
Neuron 13 - Last Output: -0.35003, Threshold: 0.591376, Connections: (Neuron 0, Strength: 0.880273) (Neuron 1, Strength: -0.501611) (Neuron 6, Strength: -0.337898) (Neuron 10, Strength: 0.489975) (Neuron 12, Strength: -0.443709) (Neuron 17, Strength: 0.880776) (Neuron 18, Strength: 0.496994) (Neuron 0, Strength: -0.709754) (Neuron 1, Strength: -0.304489) (Neuron 3, Strength: -0.328104)
Neuron 14 - Last Output: -0.322582, Threshold: -0.0722551, Connections: (Neuron 0, Strength: -0.823386) (Neuron 2, Strength: 0.837892) (Neuron 4, Strength: 0.831599) (Neuron 7, Strength: -0.0780512) (Neuron 9, Strength: -0.550981) (Neuron 10, Strength: -0.902546) (Neuron 11, Strength: -0.693969) (Neuron 15, Strength: 0.485928) (Neuron 17, Strength: 0.75341) (Neuron 19, Strength: -0.837886)
Neuron 15 - Last Output: -0.803729, Threshold: 0.655867, Connections: (Neuron 2, Strength: 0.861504) (Neuron 3, Strength: 0.398648) (Neuron 4, Strength: 0.858052) (Neuron 16, Strength: 0.267516) (Neuron 18, Strength: -0.434989) (Neuron 0, Strength: 1.03612) (Neuron 4, Strength: 0.532029) (Neuron 7, Strength: 0.533262) (Neuron 12, Strength: -0.482431) (Neuron 17, Strength: -0.363601)
Neuron 16 - Last Output: 0.294443, Threshold: 0.125319, Connections: (Neuron 0, Strength: -0.217536) (Neuron 1, Strength: 0.108554) (Neuron 2, Strength: 0.811069) (Neuron 5, Strength: -0.809592) (Neuron 10, Strength: 0.884472) (Neuron 0, Strength: -0.424622) (Neuron 2, Strength: -0.927549) (Neuron 3, Strength: -0.270085) (Neuron 4, Strength: -0.890802) (Neuron 0, Strength: -0.710248)
Neuron 17 - Last Output: 0.449173, Threshold: 0.332378, Connections: (Neuron 0, Strength: 0.802905) (Neuron 1, Strength: 0.793987) (Neuron 2, Strength: -0.958251) (Neuron 4, Strength: -0.684295) (Neuron 5, Strength: -0.225956) (Neuron 9, Strength: 0.74517) (Neuron 10, Strength: 0.436079) (Neuron 11, Strength: 0.459063) (Neuron 12, Strength: -0.994363) (Neuron 14, Strength: 0.658581) (Neuron 16, Strength: 0.881988)
Neuron 18 - Last Output: -0.109741, Threshold: 0.526719, Connections: (Neuron 1, Strength: -1.50457) (Neuron 2, Strength: 0.366154) (Neuron 4, Strength: 0.46262) (Neuron 5, Strength: -1.25867) (Neuron 8, Strength: -0.67753) (Neuron 15, Strength: -0.844055) (Neuron 1, Strength: 0.425795) (Neuron 2, Strength: 0.339732) (Neuron 6, Strength: -0.283848) (Neuron 0, Strength: -0.465414)
Neuron 19 - Last Output: 0.730713, Threshold: 0.0712403, Connections: (Neuron 0, Strength: -1.55896) (Neuron 2, Strength: 0.317348) (Neuron 7, Strength: 0.44168) (Neuron 10, Strength: -0.969468) (Neuron 17, Strength: 0.349798) (Neuron 2, Strength: -1.13351) (Neuron 4, Strength: 0.257095) (Neuron 3, Strength: 0.373297) (Neuron 1, Strength: 1.15578) (Neuron 2, Strength: 0.488853)

2. Large scale fitting test

The connection situation of the trained model after large-scale fitting testing is too complex. For detailed information, please refer to ModelConnection_100neurons. txt in the same folder as the

provided code.

3. Multi-sample test

The following is the internal connectivity of the model after multi sample testing and training.

Neuron 0 - Last Output: -0.359161, Threshold: -0.0261074, Connections: (Neuron 2, Strength: -0.567314) (Neuron 6, Strength: 0.847905) (Neuron 7, Strength: 0.683362) (Neuron 8, Strength: 0.171685) (Neuron 10, Strength: 0.19372) (Neuron 14, Strength: 0.110578) (Neuron 18, Strength: 0.123988) (Neuron 4, Strength: -0.659169) (Neuron 6, Strength: 0.283502) (Neuron 9, Strength: 0.114225)

Neuron 1 - Last Output: -0.531745, Threshold: 0.263083, Connections: (Neuron 3, Strength: 0.112132) (Neuron 4, Strength: 0.147365) (Neuron 6, Strength: -0.784508) (Neuron 9, Strength: 0.70799) (Neuron 10, Strength: -0.412091) (Neuron 11, Strength: 0.568026) (Neuron 15, Strength: 1.21658) (Neuron 18, Strength: -0.641926) (Neuron 19, Strength: 0.376614) (Neuron 0, Strength: 0.447606)

Neuron 2 - Last Output: -0.454673, Threshold: 0.646858, Connections: (Neuron 0, Strength: -0.578188) (Neuron 5, Strength: -0.618387) (Neuron 8, Strength: -0.540635) (Neuron 9, Strength: 0.127775) (Neuron 10, Strength: 0.968188) (Neuron 14, Strength: -1.00156) (Neuron 15, Strength: -1.09538) (Neuron 17, Strength: 0.168966) (Neuron 3, Strength: -0.324579) (Neuron 6, Strength: 0.64431)

Neuron 3 - Last Output: -0.220232, Threshold: 0.110508, Connections: (Neuron 1, Strength: 0.851921) (Neuron 5, Strength: -0.296442) (Neuron 6, Strength: 0.346867) (Neuron 7, Strength: 0.245091) (Neuron 10, Strength: -0.780984) (Neuron 12, Strength: 0.174535) (Neuron 17, Strength: 0.193493) (Neuron 18, Strength: 0.872823) (Neuron 6, Strength: 0.97224) (Neuron 1, Strength: -0.693607)

Neuron 4 - Last Output: -0.702331, Threshold: 0.97612, Connections: (Neuron 6, Strength: -0.870414) (Neuron 8, Strength: 0.389103) (Neuron 10, Strength: -0.338785) (Neuron 11, Strength: -0.761402) (Neuron 13, Strength: -0.837097) (Neuron 14, Strength: 0.197477) (Neuron 15, Strength: 1.02608) (Neuron 16, Strength: 0.698198) (Neuron 17, Strength: 0.617177) (Neuron 18, Strength: -0.515954)

Neuron 5 - Last Output: -0.266401, Threshold: 0.903265, Connections: (Neuron 1, Strength: -0.702469) (Neuron 6, Strength: 0.625877) (Neuron 8, Strength: -0.110889) (Neuron 10, Strength: -0.509116) (Neuron 11, Strength: -1.01309) (Neuron 12, Strength: -0.0115481) (Neuron 15, Strength: 0.19501) (Neuron 17, Strength: 0.771963) (Neuron 0, Strength: 0.317438) (Neuron 1, Strength: -0.41883)

Neuron 6 - Last Output: 0.303283, Threshold: 0.0480718, Connections: (Neuron 4, Strength: 0.119276) (Neuron 7, Strength: -0.890765) (Neuron 8, Strength: -1.10042) (Neuron 9, Strength: 0.303089) (Neuron 18, Strength: -0.526258) (Neuron 19, Strength: 0.829383) (Neuron 0, Strength: 0.834437) (Neuron 1, Strength: -0.0757071) (Neuron 2, Strength: 0.458501) (Neuron 5, Strength: 0.464773)

Neuron 7 - Last Output: 0.251497, Threshold: 0.948147, Connections: (Neuron 0, Strength: -0.384199) (Neuron 3, Strength: -0.0355585) (Neuron 5, Strength: -0.107189) (Neuron 6, Strength: 0.825463) (Neuron 8, Strength: 0.354953) (Neuron 9, Strength: -0.821468) (Neuron 10, Strength: -0.952222) (Neuron 11, Strength: -0.559735) (Neuron 12, Strength: 0.813509) (Neuron 17, Strength: 0.557956)

Neuron 8 - Last Output: 0.738281, Threshold: -0.205357, Connections: (Neuron 5, Strength:

0.741432) (Neuron 9, Strength: 0.249263) (Neuron 12, Strength: 1.09449) (Neuron 13, Strength: 0.872799) (Neuron 15, Strength: -0.826731) (Neuron 1, Strength: -0.69256) (Neuron 6, Strength: 0.323749) (Neuron 7, Strength: 0.813959) (Neuron 2, Strength: 0.0415997) (Neuron 4, Strength: -0.749815)
Neuron 9 - Last Output: -0.650102, Threshold: 0.0753206, Connections: (Neuron 3, Strength: 0.4388) (Neuron 4, Strength: 1.14081) (Neuron 5, Strength: -0.876972) (Neuron 6, Strength: 0.883659) (Neuron 10, Strength: 0.715975) (Neuron 12, Strength: -0.560736) (Neuron 16, Strength: 0.372294) (Neuron 0, Strength: -0.927758) (Neuron 7, Strength: 0.76348) (Neuron 8, Strength: -1.17943)
Neuron 10 - Last Output: -0.62609, Threshold: 0.46109, Connections: (Neuron 1, Strength: 0.101321) (Neuron 4, Strength: 0.754579) (Neuron 6, Strength: 1.09497) (Neuron 8, Strength: 0.42902) (Neuron 9, Strength: 0.584447) (Neuron 11, Strength: 1.06296) (Neuron 12, Strength: -0.46192) (Neuron 13, Strength: 0.0199342) (Neuron 16, Strength: -0.370825) (Neuron 17, Strength: 0.252809) (Neuron 19, Strength: -0.7339)
Neuron 11 - Last Output: 0.563617, Threshold: 0.0205643, Connections: (Neuron 0, Strength: -0.0675141) (Neuron 2, Strength: -0.557318) (Neuron 5, Strength: 0.186034) (Neuron 7, Strength: -0.582963) (Neuron 8, Strength: 0.974434) (Neuron 10, Strength: 0.471409) (Neuron 13, Strength: 0.133505) (Neuron 15, Strength: -0.711412) (Neuron 16, Strength: 0.0471714) (Neuron 0, Strength: -0.643166)
Neuron 12 - Last Output: -0.552218, Threshold: 0.000520035, Connections: (Neuron 0, Strength: -0.369483) (Neuron 5, Strength: 0.307384) (Neuron 6, Strength: 0.275303) (Neuron 7, Strength: -0.881617) (Neuron 8, Strength: -0.132712) (Neuron 9, Strength: -0.14686) (Neuron 13, Strength: -0.634051) (Neuron 15, Strength: 0.444534) (Neuron 16, Strength: 0.776517) (Neuron 17, Strength: -0.804091) (Neuron 18, Strength: -0.871269)
Neuron 13 - Last Output: -0.0926221, Threshold: -0.614197, Connections: (Neuron 1, Strength: 0.717318) (Neuron 2, Strength: 0.291509) (Neuron 4, Strength: -0.100611) (Neuron 6, Strength: -0.310554) (Neuron 8, Strength: 0.397127) (Neuron 11, Strength: -0.672586) (Neuron 14, Strength: -0.798384) (Neuron 15, Strength: 0.00960307) (Neuron 18, Strength: -0.118736) (Neuron 0, Strength: -0.852188)
Neuron 14 - Last Output: -0.218843, Threshold: 0.10199, Connections: (Neuron 1, Strength: 0.41679) (Neuron 3, Strength: 0.889317) (Neuron 5, Strength: 0.960944) (Neuron 6, Strength: -0.752179) (Neuron 9, Strength: -0.597483) (Neuron 10, Strength: -0.976917) (Neuron 17, Strength: 0.297063) (Neuron 19, Strength: -0.407454) (Neuron 0, Strength: 0.465785) (Neuron 3, Strength: -0.420812)
Neuron 15 - Last Output: 0.128061, Threshold: 0.14612, Connections: (Neuron 0, Strength: -1.03695) (Neuron 1, Strength: -0.200815) (Neuron 4, Strength: 0.852199) (Neuron 6, Strength: -0.399411) (Neuron 12, Strength: -0.334596) (Neuron 13, Strength: -0.518207) (Neuron 17, Strength: 0.980035) (Neuron 19, Strength: 0.527391) (Neuron 0, Strength: -0.851198) (Neuron 1, Strength: 0.00906111)
Neuron 16 - Last Output: -0.700669, Threshold: 0.618646, Connections: (Neuron 1, Strength: -0.307298) (Neuron 2, Strength: 0.236341) (Neuron 4, Strength: 0.649204) (Neuron 7, Strength: -0.414778) (Neuron 9, Strength: -0.447759) (Neuron 15, Strength: -0.00242695) (Neuron 17, Strength: 0.599801) (Neuron 19, Strength: -0.311885) (Neuron 2, Strength: -0.39036) (Neuron 4, Strength: 1.00008)

Neuron 17 - Last Output: -0.837791, Threshold: 0.216484, Connections: (Neuron 1, Strength: 0.373206) (Neuron 4, Strength: -0.317242) (Neuron 7, Strength: -0.957783) (Neuron 11, Strength: -0.366169) (Neuron 13, Strength: -0.567948) (Neuron 14, Strength: 0.299946) (Neuron 18, Strength: -0.916863) (Neuron 19, Strength: -0.460981) (Neuron 4, Strength: 1.24454) (Neuron 0, Strength: 0.597025)
Neuron 18 - Last Output: 0.618424, Threshold: 0.49317, Connections: (Neuron 6, Strength: -1.49826) (Neuron 19, Strength: -1.26173) (Neuron 3, Strength: -2.01375) (Neuron 4, Strength: -1.30685) (Neuron 0, Strength: -1.35302) (Neuron 1, Strength: -1.18628) (Neuron 0, Strength: -0.886131) (Neuron 2, Strength: -0.0881287) (Neuron 1, Strength: -0.404159) (Neuron 2, Strength: -0.508537)
Neuron 19 - Last Output: -0.0521245, Threshold: -0.518147, Connections: (Neuron 4, Strength: -0.424017) (Neuron 5, Strength: -0.933456) (Neuron 6, Strength: 0.0811713) (Neuron 7, Strength: -0.585298) (Neuron 12, Strength: 0.771161) (Neuron 15, Strength: -0.495436) (Neuron 16, Strength: 0.559586) (Neuron 0, Strength: 0.525478) (Neuron 7, Strength: 0.161966) (Neuron 10, Strength: 0.0271596)